Fast Statistical Static Timing Analysis Using Smart Monte Carlo Techniques

Vineeth Veetil, Kaviraj Chopra, David Blaauw, *Senior Member, IEEE,* and Dennis Sylvester, *Fellow, IEEE*

***Abstract*—In this paper, we propose a stratification+hybrid quasi Monte Carlo (SH-QMC) approach to improve the efficiency of Monte Carlo-based statistical static timing analysis (SSTA) using sample size reduction. Sample size reduction techniques proposed in the literature exhibit a tradeoff between accuracy of the Monte Carlo estimate with fewer samples and their ability to handle large number of variables in multidimensional space. This paper proposes to target several such techniques to different sets of process variation variables by using information about the importance of these variables to the circuit delay, and the capability of the techniques to handle multiple dimensions. Simulations on benchmark circuits up to 90 K gates show that the proposed method requires up to 224 samples for varying levels of process variation to achieve accurate timing estimates. Results also show that when SH-QMC is performed with multiple parallel threads on a quad-core processor, the approach is faster than traditional SSTA with comparable accuracy. When the proposed SH-QMC technique is supplemented with a graph pruning method the runtime is further reduced by 46–48% on average. The technique is also extended to include an incremental approach to recompute a percentile delay metric after engineering change order.**

***Index Terms*—Algorithms, computer-aided design (CAD), Monte Carlo, statistical timing, variance reduction, verification.**

I. Introduction

PROCESS PARAMETER variations have taken on increasing importance in nanometer-scale CMOS. Rather than using simple corner models that capture worst-case behavior at the device level (and lead to large guard bands), modern computer-aided design tools are moving toward a more probabilistic view of circuit timing behavior. In replacing corner models, there are two primary approaches that incorporate process parameter uncertainty in timing analysis. The first is to perform statistical static timing analysis (SSTA) by modeling gate delay as a function of process parameters and propagating these distribution functions to compute the distribution of circuit delay [1], [2]. We refer to these approaches as traditional SSTA. In traditional SSTA it has proven challenging to efficiently model skewness in the arrival time distribution which results from non-linearity of the gate delays and the maximum function. References [1]–[11] attempted to address these issues.

The second approach is Monte Carlo based SSTA, which involves selection of samples of the process variation space to obtain statistical distributions of circuit timing behavior. The application of Monte Carlo (MC) for statistical timing was discussed in Scheffer [12], where it was shown that Monte Carlo based SSTA is accurate even in scenarios with high dimensionality and non-standard distributions in the process variation space, where traditional SSTA has difficulties. However, there are two main difficulties with this approach. First, the standard MC approach of random selection of samples in the process variation space requires too many samples for sufficient accuracy, resulting in high runtime cost. Second, there is no work to show the applicability of MC based SSTA for incremental statistical timing analysis. In this paper, we address both concerns.

Standard techniques to reduce the sample size for MC based approaches exist in statistics literature and are called variance reduction techniques. The application of these techniques for parametric yield estimation has been analyzed in the literature [13]–[17], [19]–[23]. In [13], a Latin hypercube approach for parametric yield estimation is proposed. In [14], mixture importance sampling for statistical SRAM design and analysis is proposed. The approach in [15] uses the control variates technique in conjunction with importance sampling for timing yield estimation. However, while several approaches are reviewed, no results are presented. In [16], the authors proposed to use quasi Monte Carlo analysis for yield estimation. This approach cannot be directly extended to systems with large number of dimensions (variables) which is often the case with process variation. In [17], the authors addressed this issue by reducing the problem dimension using a Karhunen–Loeve expansion model of spatial correlation. The proposed problem formulation considers a grid-less spatial correlation model with assumptions of continuity, positive definiteness, and bounded variance. The results show significant speedups in terms of sample size reduction. One drawback, however, is that it is not clear if existing design flows that employ a grid-based spatial correlation model can use the properties of stochastic processes with a covariance kernel [18] while also achieving a significantly reduced set of variables that can be handled by quasi Monte Carlo (QMC). Our earlier work in

Manuscript received May 11, 2010; revised October 2, 2010; accepted December 9, 2010. Date of current version May 18, 2011. This paper was recommended by Associate Editor S. Vrudhula.

V. Veetil is with Synopsys, Inc., Mountain View, CA 94085 USA (e-mail: veetil@synopsys.com).

K. Chopra is with Mentor Graphics, San Jose, CA 95131 USA (e-mail: kaviraj_chopra@mentor.com).

D. Blaauw and D. Sylvester are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: blaauw@eecs.umich.edu; dennis@eecs.umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2011.2108030

[12] proposes to combine QMC and LHS to address the issue of QMC's inability to handle high dimensionality. In [20], the authors presented a robust theoretical framework incorporating QMC and LHS-based methods to speed up statistical timing analysis. Further, they proposed techniques to generate QMC samples tuned for optimal performance in SSTA. Other techniques proposed recently focus on SRAM designs and rare event analyses. The authors in [22] proposed to reduce the evaluation of timing statistics in the complex, but structured SRAM designs to a single chain of component circuits. A spherical importance sampling method is then employed to evaluate the simplified model. In [23], a Markov chain Monte Carlo technique is proposed for accurate estimation of the right hand tail of delay distribution. The technique is shown to be effective for rare event analyses. There have also been attempts to parallelize Monte Carlo based methods for SSTA [24], [25].

Engineering change order (ECO) and synthesis tools require incremental timing analysis techniques for fast recomputation of circuit delay with small changes in the design. To meet time to market, designers need tools capable of performing fast incremental timing analysis, and such tools need to incorporate process variations. While incremental techniques for traditional SSTA exist in the literature [1], the lack of such techniques has been a major drawback for MC based approaches to SSTA. We address the specific problem of recomputing a percentile delay metric after incremental circuit sizing. To the best of our knowledge, this paper is the first to address incremental timing analysis in MC based SSTA.

This paper has several main contributions. First, we introduce an approach for variance reduction in MC-based SSTA, stratified sampling+hybrid quasi Monte Carlo (SH-QMC). In SH-QMC, we propose to use circuit timing criticality information for sample size reduction. We use information about the criticality of variables to the circuit delay to order them. For the most critical variables, we then employ techniques that achieve high accuracy with few samples using previously known mathematically derived sequences known as low discrepancy sequences (LDSs). For the less critical variables, we use techniques that are effective for problems of higher dimensionality. The proposed approach is implemented and tested on benchmark circuits with sizes up to 90 000 gates. In general SH-QMC requires fewer than 224 samples to achieve target accuracy on the benchmarks studied for varying levels of process variation. Our results also show that the number of samples required does not increase with the number of gates in the circuit. Additionally, when SH-QMC is implemented with multiple threads on a quad core processor, it is faster than traditional SSTA for comparable accuracy. We also observe that the performance of SH-QMC scales better than traditional SSTA with circuit size.

Second, we propose a technique to recompute a percentile delay metric after incremental circuit sizing, where individual gates are resized. In this technique, we use information local to the resized gate to prune out most of the samples, leaving only a few samples to be reevaluated. Our results for the incremental computation of the 95th percentile and 99th percentile delays of benchmark circuits show that on average only 1.4% and 0.7% of original samples need to be evaluated for exact recomputation, even after sample size reduction using SH-QMC.

This paper includes significant additions to our related work in [19]. Different techniques for ordering critical variables for mapping to LDSs are evaluated, and the relationship between the number of critical variables mapped to LDSs and accuracy is analyzed. We also propose a novel graph reduction technique to additionally improve the performance of MC-based SSTA. A learning-based graph reduction approach is introduced, where a small set of SH-QMC samples is used to identify the critical nodes in the graph. This enables fast evaluation of the remaining samples and enables up to 73% additional reduction in runtime.

This paper is organized as follows. Section II discusses the applicability of existing variance reduction approaches in statistics to the statistical timing analysis domain. Section III presents our work on variance reduction for MC-based SSTA and proposes a graph pruning method to improve the efficiency of SH-QMC. Section IV proposes an approach to incremental statistical timing analysis. We present detailed results in Section V and conclude in Section VI.

II. Variance Reduction Approaches for Statistical Timing

MC-based statistical timing involves selecting samples of the process variation space to obtain statistical distributions of circuit delay. This is mapped to the standard mathematical problem of MC, which is to estimate the integral of a function, using samples in its domain. There are standard techniques for variance reduction of MC, which include quasi Monte Carlo techniques, Latin hypercube sampling, stratified sampling, importance sampling, and control variates. In this section, we briefly discuss their applicability to the statistical timing analysis framework.

A. Quasi Monte Carlo

The standard MC method addresses the problem of approximating the integral of a function $f(x)$ over the s-dimensional hypercube $C^s = (0, 1)^s$, where x represents a point in an s-dimensional space. The MC estimate of the integral f is given by the arithmetic mean of f_i, which are values of the function $f(x)$ evaluated at n samples distributed throughout the hypercube. The Koksma–Hlawka inequality relates the error bound of a method to numerically estimate an integral using a sequence of samples, to a mathematical measure of uniformity for the distribution of the points, called "discrepancy" [27] This inequality suggests that we should use a sequence with the smallest possible discrepancy to evaluate the function in order to achieve the smallest possible error bound. Such sequences constructed to reduce discrepancy are called LDSs. Quasi Monte Carlo techniques are characterized by their use of LDSs to generate samples. LDSs are deterministic sequences, in other words there is no randomness in their generation. Intuitively, these sequences are well dispersed through the domain of the function, minimizing any gaps and/or clustering of points. Fig. 1 illustrates that quasi random sequences generate samples with lower discrepancy compared to pseudo random

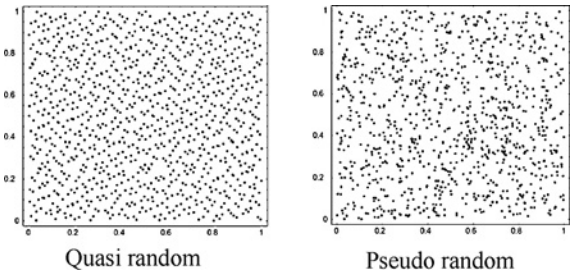

Fig. 1. Quasi random and pseudo random sequences.

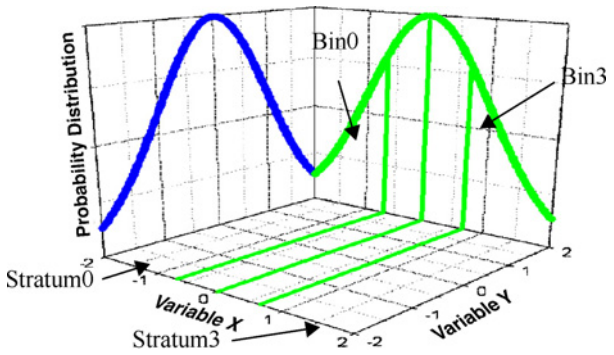

Fig. 2. Stratification of a 2-D space. Variable X is divided into four bins, thus dividing the sample space into four strata.

sequences (sequences with properties similar to "truly" random sequences). Sobol [29] and Faure and Niederreiter [16] are LDSs that have been studied extensively. In this paper, we consider Sobol sequences, which are known to be simple to construct and more resistant to the *pattern dependency* issue (mentioned below), compared to the other sequences. Interested readers can refer to [29] for a construction of the Sobol sequence, and [30] for an implementation.

In the context of statistical timing analysis, quasi Monte Carlo techniques have been studied in [16]. The author noted that LDSs are imperfect and as the number of dimensions in the problem increases, there is degraded uniformity. This effect is especially significant among the higher coordinates of LDSs, which show undesirable patterns as opposed to the low discrepancy pattern in Fig. 1. This phenomenon is referred to as *pattern dependency*. The author suggested that in timing analysis the lower coordinates of Sobol sequences, which have no significant pattern dependences, be assigned to the important variables in the sampling procedure. Therefore, a concept of criticality of variables in timing analysis needs to be defined, which can be used to sort the variables in the order of their decreasing importance. The coordinates of the Sobol sequence can then be assigned to variables in this order. We present a technique for ordering the variables based on their criticality to circuit delay in the statistical timing framework.

A related point is that Sobol sequences are not accurate beyond a certain number of dimensions. Hence, in this paper, we use quasi Monte Carlo techniques in conjunction with stratified sampling and Latin hypercube sampling (LHS). The next two subsections provide a brief overview of stratified sampling and LHS.

B. Stratified Sampling

Stratified sampling is a technique to partition the sample space into mutually exclusive strata, and then sample using any of the known variance reduction techniques within each [28]. The stratification method in this paper is illustrated for a 2-D example in Fig. 2, where random variable X is divided into four equal probability bins (X is equally likely to fall in any of the four bins), whereas random variable Y is not binned. This method is adopted when X is critical to the function value to be estimated, whereas Y is not. In this way, the 2-D space is partitioned into four strata as shown in the figure. Throughout the work, we use "bin" to refer to regions in individual variables, and "strata" to refer to partitions in the nD space, where n is the dimensionality. In general in multidimensional space, one or more variables are binned, and the permutations of bins across variables define strata. In the case of timing analysis, the timing behavior of the circuit is more sensitive to the critical variables by selection and these variables are binned. Therefore within strata the timing behavior exhibits lower variation and is easier to estimate. The technique leads to accuracy with few samples, however cannot be used over very large dimensions since the number of strata increases exponentially.

C. Latin Hypercube Sampling

Latin hypercube sampling is a technique in variance reduction which deals with multidimensional systems [31]. This technique tries to sample each variable involved uniformly by dividing the variable into equal probability bins. The samples from bins in variables are combined across dimensions to obtain faster convergence than random sampling. This is in contrast with taking all permutations of the bins across variables to define strata, and then sampling within each stratum as in stratified sampling described above. This means that LHS can deal with large dimensions, however with a moderate rate of convergence compared to full stratification.

The LHS procedure is illustrated in Fig. 3. Each random variable is divided into equal probability bins. One sample is generated within each bin. Such samples are combined across variables to obtain Latin hypercube samples. This is the procedure to obtain k samples, where k is the number of bins per variable. To obtain mk number of samples, we repeat the LHS procedure m times.

Two other techniques that have been studied for application to integrated circuit yield estimation are importance sampling and control variates. In general, these methods require more detailed information about the circuit. For literature in statistics about the method, refer to [28]. More work is required to establish the effectiveness of these approaches for use in the modern integrated circuit design process.

III. Smart Sampling Based on Timing Criticality

In this section, we first describe our process variation model and then go on to discuss our smart sampling approach.

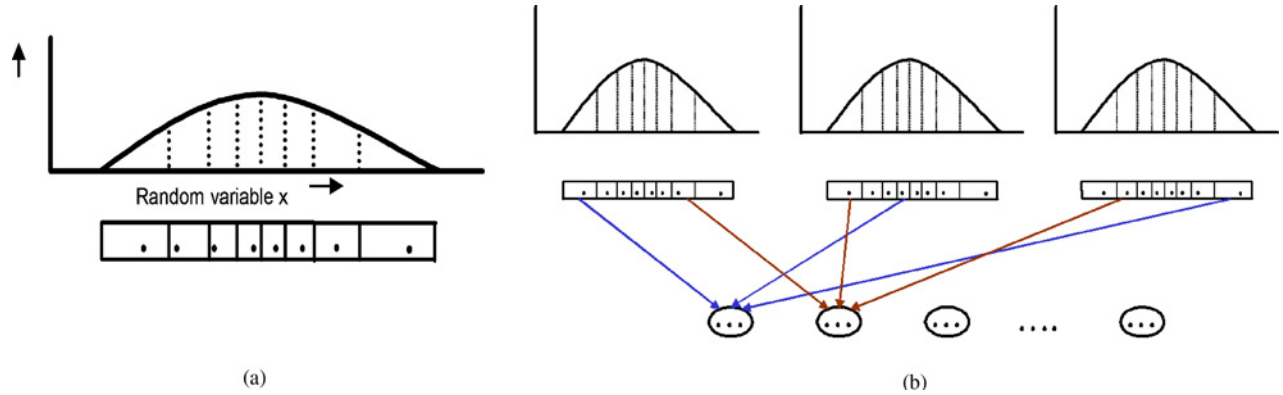

Fig. 3. Latin hypercube sampling. (a) Divide each variable in eight equal probability bins and sample in bins. (b) Combine randomly to form eight triplets.

A. Process Variation Model

Our process variation model is based on [2] which takes into account within-die (WID) spatially correlated variation by partitioning the die into $n * n$ grids and assuming identical parameter variations within a grid. Therefore, each source of variation is represented by a set of random variables for all grids. For example, transistor gate length variation is represented by a set of random variables for all grids and the set is of multivariate normal distribution with a covariance matrix R_{Lg}. Principal component analysis is performed on these correlated random variables to obtain a set of principal components. Similarly, principal components are obtained for other sources of variation. Let p_i: $i = 1, ..., m$ be the principal components of all global sources of variation. In addition to these global sources of variation, we have an independent random variable Δr to account for random variation at the gate level. The delay for a gate is expressed as a linear combination of principal components of p_i and Δr

$$l = d_0 + k_1 \times p_1 + \ldots + k_m \times p_m + k_{m+1} \times \Delta r \qquad (1)$$

where d_0 is the gate delay mean, k_i: $i = 1, ..., m$ are the coefficients for the principal components. p_i and Δr are independent unit normal random variables after suitably scaling their coefficients.

B. Stratification+Hybrid Quasi Monte Carlo (SH-QMC)

In our smart sampling approach SH-QMC, we propose to use circuit timing criticality information to reduce the sample size for MC-based statistical timing analysis. In the previous subsection, we have defined the variables representing process parameter variation. In our proposed approach, we order these variables based on their criticality to the circuit delay using a timing criticality parameter P_{crit} defined in the next subsection. We then apply QMC, stratified sampling and LHS to variables based on their convergence property and the ability to handle multiple variables (dimensions) as illustrated in Fig. 4 The topmost critical variables guide the stratified sampling approach, which leads to faster convergence. Only the top two to five variables are used to guide stratification since the number of strata increases exponentially with the number of variables as explained in Section II-B. QMC is then employed on the topmost to moderately critical variables

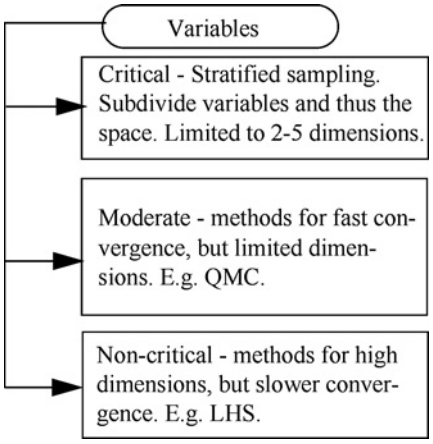

Fig. 4. Ordering variables using timing criticality.

for its fast convergence properties. However, QMC can exhibit pattern dependencies with large number of variables, so only a limited number of variables are sampled using QMC. On the non-critical variables we use Latin hypercube sampling, which is applicable for large number of variables, but has slower convergence to an accurate result.

The method is illustrated in Fig. 5 using a 5-variable example. As mentioned before, variables are ordered as critical, moderately critical and non-critical. The two most critical variables $r1$ and $r2$ are divided into four bins each [Fig. 5(a)]. A stratum is defined as a set of points in the 5-D space restricted to one bin each in $r1$ and $r2$, but unrestricted in $r3$, $r4$, and $r5$. The total number of strata is 16, arising from 4 by 4 permutations of the bins. Fig. 5(b) illustrates one particular stratum which we use to explain the remaining steps. In this stratum, points are restricted to bin 2 in $r1$ and bin 3 in $r2$. As shown in Fig. 5(c), QMC method based on Sobol sequence is used to sample $r1$, $r2$, and $r3$ in the stratum and LHS is applied to $r4$ and $r5$. Note that since we are only sampling within the stratum, samples of r1 and r2 are restricted to the respective bins. QMC generates triplets as shown in the figure. For performing LHS, $r4$ and $r5$ are divided into eight bins each and one value is selected from each bin as in Fig. 5(c). Eight LHS pairs are generated by randomly picking from $r4$ and $r5$ in one step of LHS. Two LHS pairs are shown in Fig. 5(d). Next, the LHS pairs are combined with the QMC triplets to

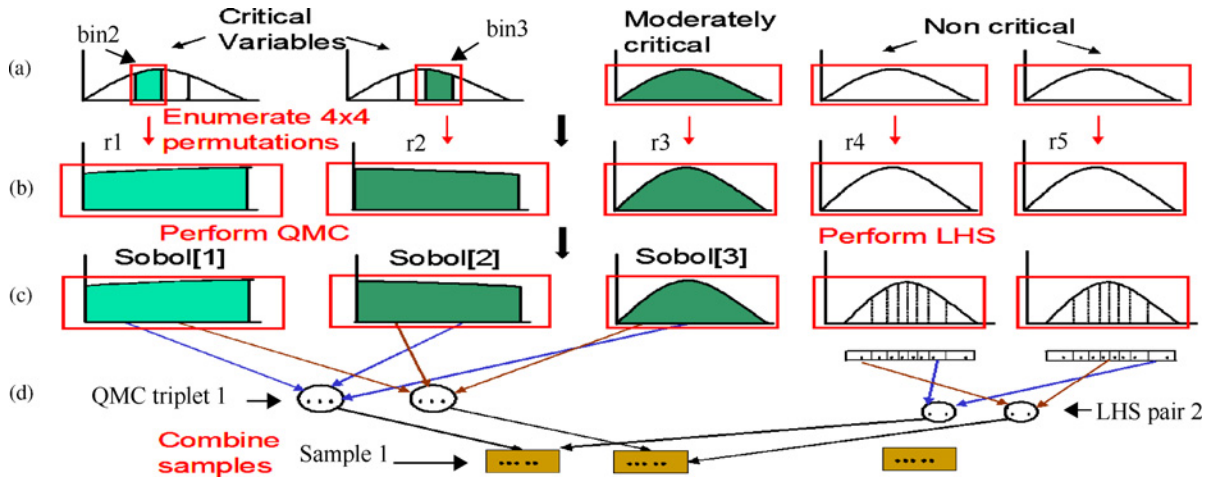

Fig. 5. Stratified Latin hypercube sampling. (a) Ordering variables based on timing criticality. (b) One of 16 strata in the sample space. (c) QMC triplets and LHS pairs. (d) These are combined to obtain final samples.

generate our final samples. The procedure is repeated: LHS pairs are generated again in $r4$ and $r5$, and QMC triplets are generated in the other three variables. These are then combined as before. After generating the samples in this stratum, we move to the next stratum and repeat our steps. In this manner, we generate samples in all 16 strata.

As mentioned in quasi Monte Carlo, among the variables on which QMC is employed, the lower coordinates of LDSs are assigned to the more critical variables. The order of criticality here is again decided using the parameter P_{crit}.

We investigated the impact of the number of critical variables mapped to LDS sequences (employed in QMC sampling) on the accuracy of SH-QMC. Fig. 6 shows the 95th percentile of the error distribution (expressed in percentage) in estimating the σ of the arrival time distribution of the benchmark circuits studied (compared to a golden Monte Carlo analysis with 40k samples) with respect to the number of critical variables sampled using QMC. Results are shown for all the benchmark circuits studied. Based on this error metric, it can be inferred that the technique provides estimates of circuit timing variance within $\sim 5\%$ when 20 or more of the most critical variables are sampled using QMC. This analysis is used to guide the choice of the number of critical variables mapped to LDS sequences in our technique.

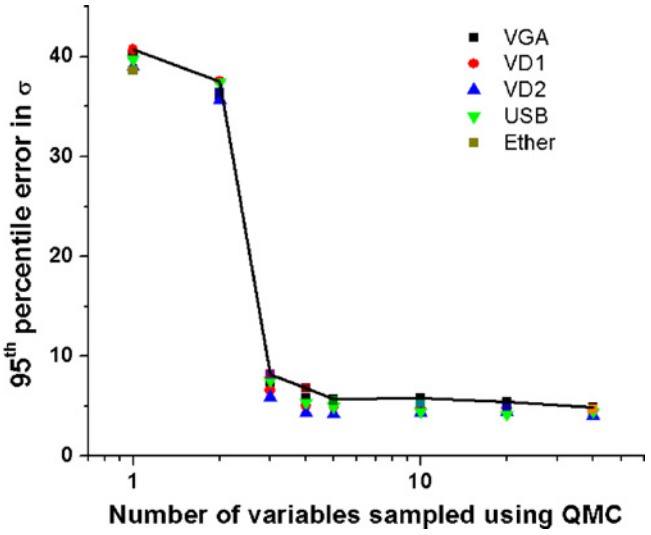

Fig. 6. Error in estimating ρ versus number of variables that are assigned quasi Monte Carlo samples.

C. Variable Ranking Based on Timing Criticality

As mentioned in Section III-B, process variation variables are ordered based on their importance or criticality to circuit timing behavior. This information enables application of QMC, LHS, and stratified sampling based on the importance of the variables. For example, QMC methods have fast convergence but can only handle limited number of variables, so the top few variables are sampled using this method. Therefore, the ordering of variables based on criticality has a direct impact on accuracy of the smart sampling technique. This section compares two different heuristics to order principal components. These techniques are as follows.

1) *Nominal ordering*. This heuristic uses information from STA performed at the nominal process corner to order variables based on a timing criticality metric P_{crit}. Intuitively, this metric prefers variables or principal components that have a higher correlation with process parameter variation in grids having more pronounced impact on overall circuit criticality behavior. The importance of grids in turn is heuristically computed as the sum of near-critical gates falling in the grid. A gate is near-critical if it has a slack of less than $s\%$ of worst-case arrival time, where "s" is a parameter. The computation of P_{crit} is performed as follows. As mentioned, each grid is assigned a weight equal to the number of gates falling in any of the potential critical paths. Let $W(i)$ be the weight of the ith grid. The weight of the jth principal component is given by

$$w_j = \Sigma_l (W(i) \times k_{ij}) \tag{2}$$

where k_{ij} is the coefficient of the jth principal component in the ith grid variation. Variables are then sorted

TABLE I
COMPARISON OF ERROR FOR *NominalOrdering* AND *Learning-Based Ordering* TECHNIQUES TO RANK VARIABLES BASED ON TIMING CRITICALITY

Circuit	SH-QMC Sample Size	Nominal Ordering (%)	Learning-Based Ordering (%)
VD1	112	4.4	4.7
VD2	112	4.1	3.7
USB	128	4.6	3.8
Ether	160	4.7	4.5
VGA	112	4.9	5.1

The error shown is the 95th percentile relative error in estimating σ of the circuit delay (in %).

based on P_{crit}. A higher value of P_{crit} corresponds to higher criticality for the variable. This expression for weight computation is restricted to the case of a single process parameter. For multiple process parameters, a more advanced expression is required as discussed at the end of this section.

2) *Learning-based ordering*. This technique builds upon the *nominal ordering* technique described above. Variables are initially ordered based on the *nominal ordering* technique. After a subset of samples S generated in the proposed SH-QMC technique are evaluated using STA, the information available is used to improve the variable ordering. The subset selection is similar to the approach presented in Section III-D, where it is discussed in more detail. At each sample in S, a variable ordering is obtained using the heuristic used in *nominal ordering*, except that near-critical gates are identified using slacks obtained at the particular sample instead of the nominal process corner. For each principal component the weights across all the samples in S are added to obtain the final weights. Principal components are sorted according to the weights to obtain the final variable ordering.

Whereas *nominal ordering* uses information at the nominal process corner, *learning-based ordering* uses information at multiple samples in the process variation space. In Table I, we compare the two techniques based on the accuracy of the SH-QMC analysis compared to a golden of Monte Carlo analysis of 40 000 randomly generated samples. As will be described in more detail in Section V, the error metric is the 95th percentile of the error in estimating σ of the worst arrival time. The first column shows the error for SH-QMC using *nominal ordering* and the second column shows the error for the case of *learning-based ordering*. The results indicate that the two techniques are comparable in accuracy for the benchmark circuits considered. Since *nominal ordering* is a simpler heuristic, this technique is used in our implementation.

For the case of multiple process parameters (each of which is resolved into principal components for die-to-die and within-die variation), the weight for the ith grid used in computation of the weight for principal component j is a function of the corresponding process parameter l. This weight is defined as the sum of sensitivities of gate delays in grid i to variation in the process parameter l at the nominal process corner, such that the gates have slack less than $s\%$ of worst-case arrival time. The sensitivity value is obtained from the statistical characterization library for the gate type. The expression for computing the weight $W_l(i)$ for grid i is therefore modified to

$$w_l(i) = \sum_g \frac{\partial d_g}{\partial l} \tag{3}$$

where g represents gates in grid i with slack less than $s\%$ of worst-case arrival time, d_g is the delay of gate g. This expression is substituted in (2) to replace $W(i)$ to obtain principal component weight w_j.

D. Critical Graph Analysis for MC SSTA

In this section, we propose a technique to improve the performance of smart sampling based SSTA through critical graph analysis. The basic idea is to identify critical paths in the graph through heuristic techniques. Gates which are expected to have a negligible effect in determining the worst case arrival time of the circuit are pruned or avoided from consideration in subsequent analyses. If the number of such gates is high, this leads to speedups in the overall statistical analysis. In the context of variability, criticality is statistical. The challenge here is to assign probability values to gates/paths in the circuit based on a measure of criticality. In [32], the authors proposed an algorithm to compute criticality probability of gates in the circuit. This algorithm computes criticality accurately, however it can potentially add a significant runtime overhead to the SSTA. It may be noted that the proposed critical graph analysis technique only requires that all sufficiently critical gates be selected for accuracy in subsequent SSTA. The exact values for criticality probability are not required in the further analysis. Therefore, we propose simpler techniques for critical graph identification. We propose that slack information obtained from a learning-based approach involving evaluation of a subset of the SH-QMC samples, be used to identify the critical graph. The timing overhead for the technique is significantly lower.

We first discuss a nominal STA-based critical graph identification approach to illustrate the graph reduction concept, before we explain the proposed learning-based approach.

1) *Nominal STA-Based Critical Graph Identification:* This technique uses information obtained from timing analysis of the circuit at the nominal process corner. The example in Fig. 7 illustrates the technique. Nominal STA is performed and slack information is obtained at all gates in the circuit. Gates with significant slack, higher than a threshold value of 0.3 units in this example, are excluded from consideration when applying MC based SSTA. The reduced graph size will allow the runtime-dominant MC STA runs to be reduced roughly linearly with circuit size. The threshold slack is defined as $s_T\%$ of the worst arrival time at the nominal sample, where s_T is the pruning parameter.

2) *Learning-Based Critical Graph Identification:* A subset of SH-QMC samples or training samples are evaluated to extract more information about the statistical behavior of the circuit. This information is used to obtain bounds on the

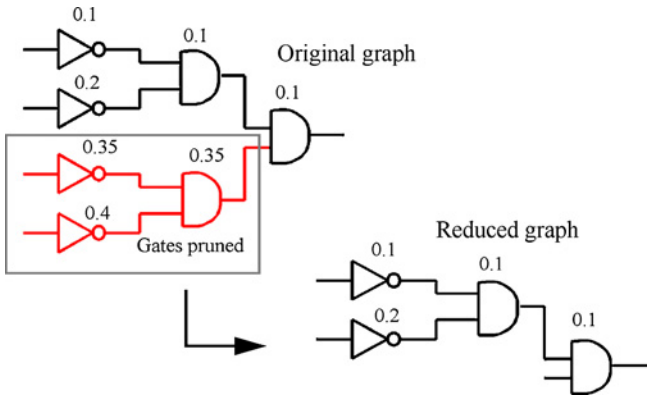

Fig. 7. Illustration of graph reduction. Slacks for nodes are indicated above corresponding gates. Gates with slack higher than 0.3 units at their output node are removed to obtain the reduced graph in this example.

probability distribution of timing slack at each node. When enough information is gathered to label a certain node as having negligible probability of lying on a critical path for any sample, the node is pruned or eliminated from consideration for remaining SH-QMC samples.

As described in Section III-B, SH-QMC combines stratified sampling, QMC sampling, and LHS. First, the sample space is partitioned into strata. Next, QMC and LHS are applied in combination within each stratum. The subset of SH-QMC samples, to be evaluated for the training, is selected such that each LHS bin has exactly one value for the corresponding variable. For example, suppose the LHS technique used divides each variable into five bins and there are four strata in the process variation space. Then the subset has 20 samples; five samples corresponding to the bin size for LHS within each stratum. Samples in this subset are evaluated, the overall idea being to ensure uniform coverage of process space. Note that QMC has no granularity unlike LHS and does not affect the training set size. At every circuit node we thus have a slack distribution obtained from the subset. A low percentile of the slack distribution is the metric considered for pruning each gate. A gate is pruned if the value is positive, in other words the probability of the gate having close to zero slack is very low in any process variation sample. Determining the optimal percentile point of slack distribution for pruning is a challenge; lower percentile points are expected to be accurate but limit runtime improvements.

The learning-based technique can be augmented by performing the nominal STA based critical graph identification approach, before the training samples are evaluated. This reduces the runtime for the training or learning-based critical graph identification step. Note that the nominal STA does not add to the runtime cost, as it is a step in the existing flow for variable ranking, discussed in Section III-C.

We refer to a pruning approach which employs only the learning-based critical graph identification approach as a single-stage pruning approach. In a two-stage pruning approach, this is augmented with nominal STA based critical graph identification. Their comparative merits are discussed in Section VI.

IV. Incremental Evaluation of a Percentile Delay

ECO and synthesis tools require efficient incremental timing analysis techniques for fast recomputation of circuit delay with small changes in the design, while also accounting for process variation. In MC based SSTA there is a lack of incremental capability to date. In this section, we present an approach for the incremental evaluation of a specific percentile delay of a circuit with a small change in circuit sizing. We illustrate the approach for the case of single gate sizing in this paper. However, the approach can be extended to the case of simultaneous multiple gate sizing. The key intuition is that if the samples for SH-QMC on circuit C are reused for C' (C with gate g sized), then most samples need not be reevaluated to recompute the xth percentile delay; only those samples that have a circuit arrival time "close" enough to the xth percentile delay of C need to be reevaluated. An upperbound on change in circuit arrival time of a sample from C to C' can be determined from a local bound computation involving only a few gates connected to the gate g being resized. This bound can be used to prune out a majority of the samples, leaving us with a few that need to be reevaluated. Further speedup can be achieved with established techniques for incremental STA on the samples selected for reevaluation.

A. Algorithm

We perform timing analysis on an original circuit C using our SH-QMC approach and store the samples for the process variation space and the corresponding circuit arrival time in memory. Our approach for the recomputation of a specific percentile delay using the stored samples is illustrated in Fig. 8. For each sample, a bound on change in circuit arrival time from C to C' (C with gate g sized) is obtained as explained in Section IV-B. Each sample has a positive bound and negative bound for either direction of change. The samples are sorted in the order of increasing circuit arrival time for C. In Fig. 8(a), the samples are represented by points on the circuit arrival time distribution curve. They are visited in the decreasing order of arrival time starting from the xth percentile value t_x. A sample k is selected for reevaluation if its arrival time for circuit C and the positive bound for k add up to exceed t_x. For example, in Fig. 8(a), sample i is pruned out since its positive bound is not large enough to cross t_x. However, sample $i-1$ is reevaluated as it has a large enough upper bound to cross t_x. As illustrated in Fig. 8(b), the arrival time for $i-1$ is recomputed. Sample $i-1$ is updated with this value of arrival time which shifts t_x to the right. Next sample $i-2$ is reevaluated, however the arrival time value obtained is less than t_x, so t_x does not change. Sample $i-2$ is also updated with the recomputed arrival time value. After considering all samples to the left of t_x, we visit the samples to the right. The criterion for reevaluating a sample here is that its arrival time for C and the negative bound for the sample should add up to less than t_x. After this step, we repeat the procedure and visit samples to the left of the updated t_x. Samples reevaluated earlier are not visited again. The termination criterion is that there are no samples to the left or right of t_x which satisfy

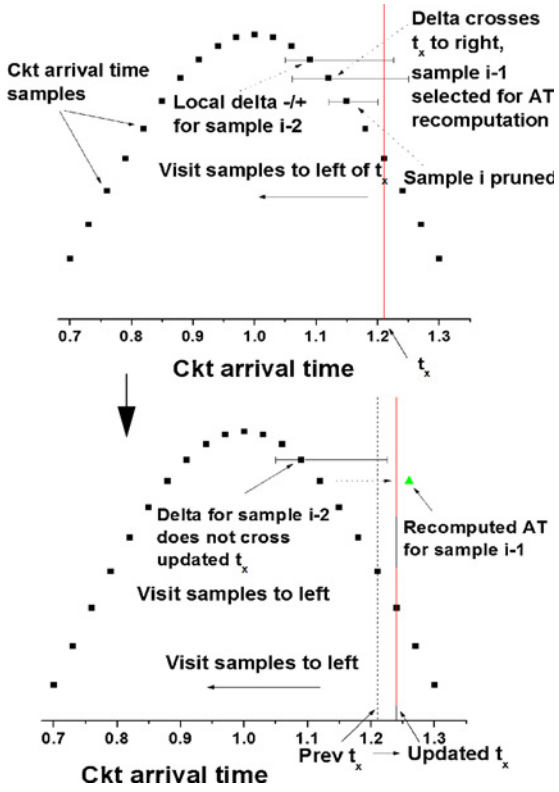

Fig. 8. (a) Samples are visited in decreasing order of circuit arrival time, starting from the xth percentile (t_x). Samples with delta crossing t_x are selected, others pruned. (b) Recomputation of circuit arrival time is performed at the selected sample and t_x is updated.

the criterion for reevaluation. The final value of t_x is the xth percentile delay of C'.

The justification for reuse of samples is that our metric to guide SH-QMC P_{crit} (Section III-C) is measured at the grid level in our process variation model, so within reasonable ECO changes the timing criticality of the circuit does not change to significantly alter our metric P_{crit}. In particular, we are only concerned about the relative ordering of variables based on P_{crit}. Therefore with single gate sizing, the samples are still accurate. For cases where there is significant design change, SH-QMC is performed again to generate new samples and the critical graph analysis step is repeated. As mentioned the samples for C are stored in memory. Our results on the benchmarks studied demonstrate that the number gof samples for SH-QMC that gives sufficient accuracy is 224 for the largest circuits. Therefore, we need to store 224 samples for each gate. Section III-A defines the variables to model process variation, which are the principal components for all sources of variation and an independent random component at the gate level. Now, it is enough to store samples for these components, as the device parameters can be retrieved using the values of components. Storing samples for the principal components incurs negligible memory overhead. In the case of the independent random component, instead of storing all samples of the component for all the gates, we store the initial "seed" value for the pseudorandom number generator. Note that for STA, gate delays are propagated in the topological order. This offset in the topological order along with the "seed" value is provided to the pseudorandom number generator which reproduces the random numbers while incremental analysis is performed. Additionally, for incremental propagation in the fanout cone of gates affected by sizing gate g, arrival time and slew values for each gate in the original circuit need to be stored for each sample. For large circuits with millions of gates, there are scalability challenges associated with such memory requirements.

B. Computing Circuit Arrival Time Bound for Samples

We compute the maximum possible increase and decrease in the circuit arrival time for each sample of circuit C using local gate delay change information when gate g is sized. Define sets *Fi(g)* of fanin gates of g, *FoFi(g)* of fanouts of gates in *Fi(g)* and *Fo(g)* of fanout gates of g. We select subpaths that are candidates for obtaining the bounds in circuit arrival time and evaluate the change in delay of these subpaths when g is sized. Every subpath starting from an input pin of a gate in *Fi(g)* and ending in an output pin of a gate in either *Fo(g)* or *FoFi(g)* is a candidate for this evaluation. Some such subpaths could have more than one gate in *Fi(g)*. We assume that delay change is significant only in the gates in the three sets defined above, therefore only these gates affect the change in subpath delay. Now, we obtain bounds for circuit arrival time change for a sample S as follows. Let *P(g)* be the set of all candidate subpaths. $t_S(p)$ and $t'_S(p)$ are delays for subpath p in sample S before and after sizing gate g, respectively. Then the negative and positive bounds are given by

$$delta\_neg(g, S) = nin\{ts'(p) - t_s(p) \forall p \in P(g),\ 0\} \quad (4)$$

$$delta\_pos(g, S) = nax\{ts'(p) - t_s(p) \forall p \in p(g),\ 0\}. \quad (5)$$

In other words, we find the maximum and minimum values of the change in delay of candidate subpaths. As gate delay change is assumed to be significant only in the local subcircuit [set of gates belonging to *Fi(g), Fo(g),* and *FoFi(g)*], the computational overhead is low. In our algorithm in Section IV-A, we only need either of *delta_neg* or *delta_pos* for most samples. A *delta_neg* or *delta_pos* computation for a sample involves gate delay computation and propagation in the local subcircuit twice, one each before and after gate sizing. Therefore, the cost of arrival time bound computation across all the samples for the percentile delay recomputation is approximately twice that of performing Monte Carlo analysis on the local subcircuit with smart samples. The runtime for this is negligible compared to that of a single STA run for most practical circuits.

V. Results

Our simulation results are based on the error in estimating statistical moments of arrival time distribution for a given method with respect to the moments from a golden of 40 000 Monte Carlo runs. Consider for example a given trial MC_1 of size 100 samples. This gives a circuit arrival time distribution. From this, moments μ_1 and σ_1 (mean arrival time and standard

deviation in arrival time) are obtained and error (magnitude of deviation from the golden) calculated for both. From repeated trials (each of 100 samples in this example), we obtain two distributions for error. The nature of the error distributions show the efficiency of the technique. For example, as we increase the number of samples from 100 to 200 in the above example and repeat the experiments, the error distribution is expected to become tighter and closer to zero. In particular, the 95th percentile of the error is closer to zero and we use this value as a criterion to compare different techniques. The minimum number of samples required by a technique such that the 95th percentile of error distribution is less than 5% for both mean arrival time and standard deviation of arrival time is our performance metric for the technique.

The number of grids in the spatial correlation model for individual circuits is varied linearly with post-placement area starting from 2×2 for the smallest circuit to 16×16 for the largest circuit. This corresponds to a grid area of approximately $40\,\mu\text{m} \times 40\,\mu\text{m}$ for all the circuits. We compare the proposed SH-QMC approach with random sampling and LHS based techniques. Simulations are performed on five large benchmark circuits. These are Viterbi Decoder 1 (VD1), Viterbi Decoder 2 (VD2), USB2.0 Core (USB), Ethernet MAC Core (ETHER), and VGA Controller Core (VGA), with gate counts varying from approximately 15 000 to 90 000. We perform synthesis and APR on all the circuits using commercial tools.

A. SH-QMC

The results are based on a 65 nm industrial technology library. In our implementation we consider channel length, oxide thickness, and threshold voltage variations as sources of process variability. The inter-die, spatially correlated within-die, and uncorrelated random components of channel length variation are considered. The relative amounts of process parameter variation among die-to-die, spatially correlated, and random sources have been studied in the literature [33]–[35]. We study the performance of the SH-QMC technique for three different process variation models as indicated in Table II. The number of samples required to achieve 95th percentile confidence in estimating mean and standard deviation of arrival time with less than 5% error is reported. Variation model A considers only channel length variation (no variation in oxide thickness and threshold voltage). An overall standard deviation of 5% is considered. The overall variation is equally divided among die-to-die, spatially correlated WID and random WID components in this model. In variation model B, all three sources (channel length, oxide thickness, and threshold voltage) are considered. The contribution among D2D, spatially correlated WID, and random WID components is the same as in model A for channel length and oxide thickness. The standard deviation for oxide thickness here is 1.3% [36]. The threshold voltage variation is modeled based on [34] where a Pelgram model is used to compute the random component of threshold voltage variation. Variation model C increases the contribution of D2D components of channel length and oxide thickness to 50% while dividing the random and spatially correlated WID components equally.

TABLE II
COMPARISON OF SAMPLE COUNTS TO ACHIEVE TARGET ACCURACY USING THE SH-QMC METHOD FOR DIFFERENT MODELS OF PROCESS VARIATION

Circuit	No. of Gates	SH-QMC Count		
		Process Variation Model A	Process Variation Model B	Process Variation Model C
VD1	14 503	160	160	112
VD2	34 082	160	128	112
USB	32 898	176	176	128
ETHER	57 327	192	208	160
VGA	90 831	224	176	112

The average number of samples required to achieve target accuracy using SH-QMC are 182, 170, and 125, respectively, for process variation models A, B, and C. Sample counts for A and B are comparable, indicating that adding more process parameters does not cause any significant increase in number of samples. Fewer than 224 samples are required to achieve target accuracy in each case. Further, there is no notable increase in sample size with respect to size of the benchmark circuits from results obtained. Results presented in the rest of this paper are based on process variation model C unless otherwise stated. Section III-C mentions that critical paths are identified within a slack of $s\%$ for computing timing criticality P_{crit}. To investigate the sensitivity of the results to the parameter s it is varied from 1–5%. Results indicate there is no change in the number of samples required to meet the stated accuracy objective, indicating that the proposed technique is stable with respect to this parameter. Table III shows the sample counts for ISCAS 85 benchmark circuits [40] and the additional benchmark circuits to achieve target accuracy using process variation model C. Additionally, the sample counts required to achieve a more stringent measure of accuracy are shown. This error metric is such that the 99th percentile of the absolute error distribution does not exceed 3%. Results indicate that fewer than 208 and 288 samples are required across the benchmark circuits to achieve the target accuracy in terms of the two error metrics.

Table IV compares the runtime of SH-QMC and an analytical SSTA model as proposed in [3], referred to as traditional SSTA in the remaining discussion. A grid-based spatial correlation model for process variation is assumed as described in Section III-A. A canonical expression for arrival time at any gate is maintained during the timing analysis, expressed as the sum of principal components representing spatial correlation and an additional variable for the within-die uncorrelated component. *Sum* operations are performed by adding the coefficients of each variable, except the random component for which the root of the squared sum of coefficients is computed. *Max* operation is approximated by matching the mean, variance and correlation of the max of random variables, as discussed in [37], while maintaining the canonical expression for the max. For both mean and standard deviation of arrival time the error for SH-QMC in the table is the average absolute deviation from their values in the golden model; for traditional SSTA this is the error with respect to the golden.

TABLE III
SAMPLE COUNTS TO ACHIEVE TARGET ACCURACY USING THE SH-QMC METHOD FOR ISCAS 85 BENCHMARK CIRCUITS [40] AND FIVE ADDITIONAL BENCHMARK CIRCUITS

Circuit	No. of Gates	SH-QMC	
		95p.c < 5%	99p.c. < 3%
c432	256	160	272
c499	544	176	288
c880	500	208	288
c1908	603	112	192
c2670	780	192	240
c3540	1163	176	256
c5315	1692	208	288
c6288	3834	208	272
c7552	2152	208	288
VD1	14 503	112	224
VD2	34 082	112	208
USB	32 898	128	176
ETHER	57 327	160	240
VGA	90 831	112	240

Process variation model C is used to generate results. The sample counts to achieve (a) <5% for the 95th percentile, and (b) <3% for the 99th percentile of the error distribution are shown.

The golden model is MC with 40 000 random samples. One drawback of Monte Carlo techniques in general is that every time an experiment is performed, the error with respect to golden is different. This means that the error in one particular MC experiment is sometimes higher than the average value mentioned. However, the 95th percentile of the absolute error distribution is still less than 5% for all the circuits in the table. This translates to an error of 3–7 ps in absolute time for different circuits. This is less than 0.4 F04 delay for this process technology [38], which is a reasonable target for the error. All our simulations were performed on an AMD 2.4 GHz Quad-Core processor operating in a Linux environment. For SH-QMC, we perform two different experiments; in one we run parallel processes on the Quad-Core machine, and in the other we run a single process on the machine. The system call *fork* is used to create child processes for the parallelization. The *pipe* system call is used for communication between each child process and the parent process. The parallelism in evaluation of samples, which are independent computations, is utilized in the implementation. Such a straightforward source of parallelism is not available in traditional/analytical SSTA based methods. We consider circuits with more than 10 000 gates for meaningful runtime comparisons. SH-QMC with multi-threading performs better than traditional SSTA in runtime. Also, further speedup in SH-QMC can be achieved in a straightforward manner using parallel processing on more than one processor. Fig. 10 compares the performance of traditional SSTA with SH-QMC for the VGA circuit as a function of number of grids in the process variation model. The number of samples at each data point in SH-QMC is selected such that the target accuracy metric is achieved in each case, as discussed earlier. The plot illustrates that SH-QMC scales better than traditional SSTA with respect to the grid complexity. The runtime for traditional SSTA approaches scales with the number of principal components which model spatial correlation effects, as discussed in Section III-A. On the other hand, for a circuit of the same complexity, the runtime for SH-QMC is proportional to the number of samples required for accurate analysis. As shown in Table II, the number of samples to achieve target accuracy does not increase significantly with increasing circuit size and grid complexity. This explains why SH-QMC scales more favorably than traditional SSTA with respect to grid complexity. Fig. 9 compares the probability distribution curve of arrival time of the USB circuit for SH-QMC (128 samples) and a traditional SSTA approach, with respect to the golden. Our technique captures the mean arrival time (marked with vertical lines) and the overall shape of the distribution better than the traditional SSTA approach.

We present results for estimation of the 99th quantile of the circuit arrival time using SH-QMC analysis in Fig. 11. Results are presented for five large benchmark circuits. The

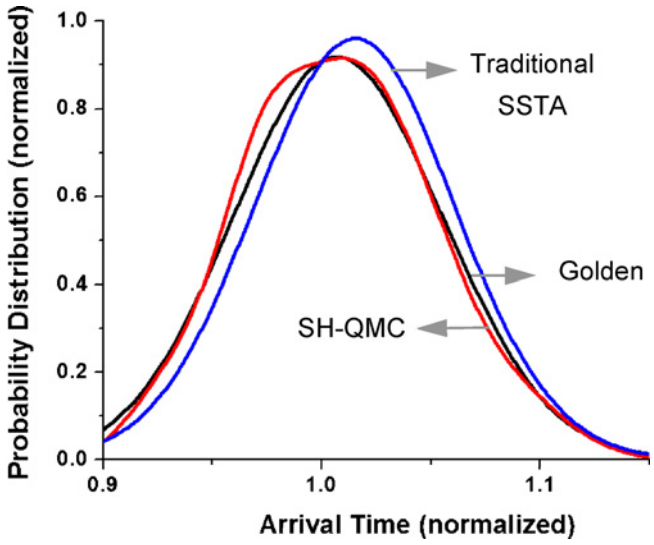

Fig. 9. Arrival time distribution of SH-QMC (128 samples) and traditional SSTA with respect to golden (40 000 MC) for USB benchmark.

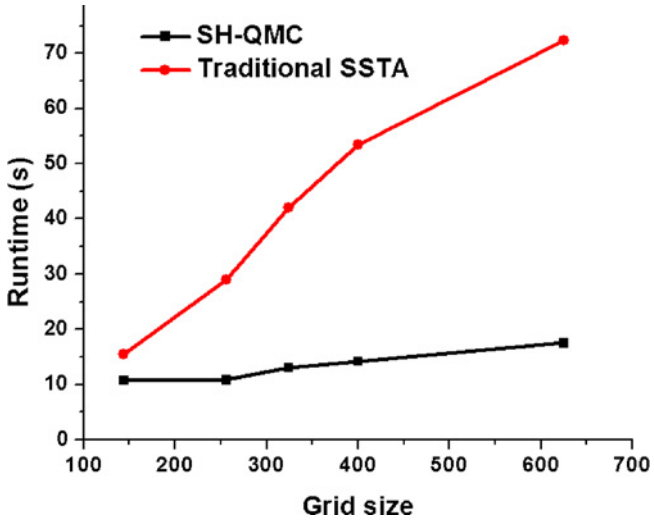

Fig. 10. Performance comparison of traditional SSTA with multithreaded SH-QMC for VGA circuit (90 831 gates) as a function of number of grids in the process variation model.

TABLE IV
RUNTIME COMPARISON OF SHQMC WITH SSTA

Circuit	No. Gates	Mean AT Error (%)		σ AT Error (%)		SSTA Run Runtime (s)	SH-QMC Runtime (s)	
		SSTA	SH-QMC	SSTA	SH-QMC		Multithread	Single-thread
VD1	14 503	0.4	0.06	0.35	1.9	2.4	1.25	3.9
VD2	34 082	0.5	0.06	1.63	1.6	14.0	4.1	13.0
USB	32 898	0.9	0.05	3.3	1.9	15.7	4.3	13.6
ETHER	57 327	0.9	0.08	4.65	1.9	27.3	10.0	32.3
VGA	90 831	0.3	0.04	0.95	2.1	53.5	14.1	44.2

AT = circuit delay.

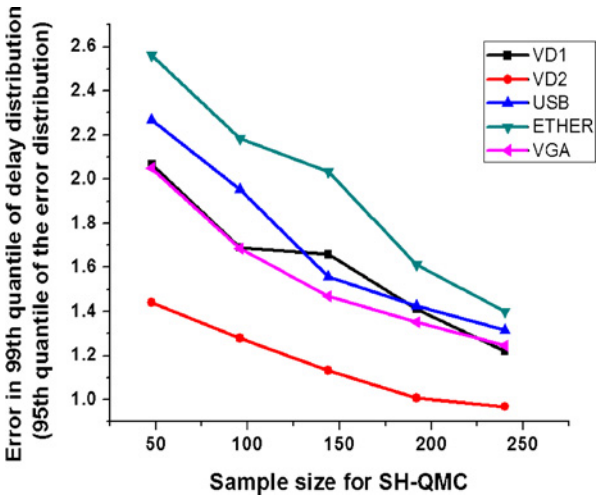

Fig. 11. Error in estimation of the 99th quantile of circuit delay distribution. Repeated trials of SH-QMC yield to a distribution of the 99th quantile value. 95th quantile of the absolute error distribution is reported. Accurate results are obtained with a small number of samples as illustrated.

error metric plotted here is the 95th percentile of the absolute error distribution compared to a golden Monte Carlo run with 40 000 samples. It is shown that high levels of accuracy are obtained (<3% error) with fewer than 50 samples across the benchmark circuits.

B. Critical Graph Identification Approach

As explained in Section III-D the pruning criterion for a gate is that a low percentile of the gate slack distribution be non-negative. The cutoff percentile is determined in terms of a pruning parameter k; if $m - ks$ of the gate slack distribution is non-negative, then the gate is pruned. For example, if k is 4 and the gate slack distribution is normal, the gate is pruned if its 0.004th percentile is non-negative. All results presented are based on a single-threaded implementation. Figs. 12 and 13 plot the error in estimation of the circuit delay s and runtime, while varying k for the pruning approach. As k increases the pruning criterion becomes more restrictive, leading to higher accuracy while degrading runtime gains. Two approaches *one-stage* and *two-stage* pruning are compared in the plots. In the two-stage pruning approach, the nominal STA based critical graph identification technique is used initially; after the training samples are analyzed, the learning-based technique is applied to further prune non-critical gates for the rest of the samples. In the one-stage approach only the learning-based approach is used (meaning the training samples are analyzed using the full circuit with no initial pruning).

The error metric is the 95th percentile error in estimating the standard deviation in arrival time compared to a golden Monte Carlo analysis with 40 000 samples. The error is compared to the SH-QMC approach without graph pruning in Fig. 12. For pruning parameter k exceeding 4, the errors are comparable for both single-stage and two-stage pruning approaches relative to the case without graph pruning. The runtimes of both pruning approaches in Fig. 13 are better than the case of no pruning since non-critical gates have been efficiently pruned from consideration after evaluating a few samples. Two-stage pruning is faster than single-stage pruning since highly non-critical gates are removed before the training samples are evaluated in the two-stage approach. In general, the errors of both single and two-stage approaches are comparable to the case of no pruning for all the benchmark circuits for pruning parameter k exceeding 4. Table V compares the three SH-QMC based approaches. The pruning parameter k is set at 4 where the errors of all three approaches are comparable for the benchmark circuits. Single-stage pruning reduces runtime by 46% and two-stage pruning by 48% on average. The learning-based critical graph analysis approach is also amenable to parallelism. Consider an SH-QMC sample size of 224. In the learning stage, 16 of the 224 samples are analyzed. For a small number of parallel machines/cores (up to 16 in this case), parallelism in the training sample evaluation stage is not affected since sufficient sample parallelism is available in this stage. Also, the sample evaluation steps (including training sample evaluation as well as evaluation of the remaining samples) dominate the overall runtime cost as indicated in Table VI. The parallel implementation is also not affected by an increase in circuit size or number of principal component variables based on similar arguments.

We demonstrate the accuracy of the critical graph identification approach in Fig. 14 for benchmark VD2 using a two-stage pruning approach with $k = 4$. After the pruning step is performed, 80k samples are generated and circuit delay computed at each sample for the critical graph. The error for each sample is then obtained in comparison to analysis of the full graph (without pruning). This error is plotted in the x-axis. The y-axis shows the number of samples (80 k total) with a given range of error. 99.6% of samples incur no error due to critical graph analysis. Our experiments show that a similar level of

TABLE V
COMPARISON OF THREE SH-QMC-BASED APPROACHES WITH NO PRUNING, 1 STAGE PRUNING, AND 2 STAGE PRUNING ON BENCHMARK CIRCUITS

Circuit	No. of Gates	No Pruning	Single Stage		Two Stage	
		Runtime (s)	% Gates Pruned	Runtime (s)	% Gates Pruned	Runtime (s)
VD1	14 503	3.9	65.3	2 2	67.3	2.1
VD2	34 082	13.0	39.4	10.1	40.7	10.0
USB	32 898	13.6	70.3	6.9	76.1	6.2
ETHER	57 327	32.3	97.1	8.9	97.3	8.8
VGA	90 831	44.2	63.5	25.2	66.0	24.2

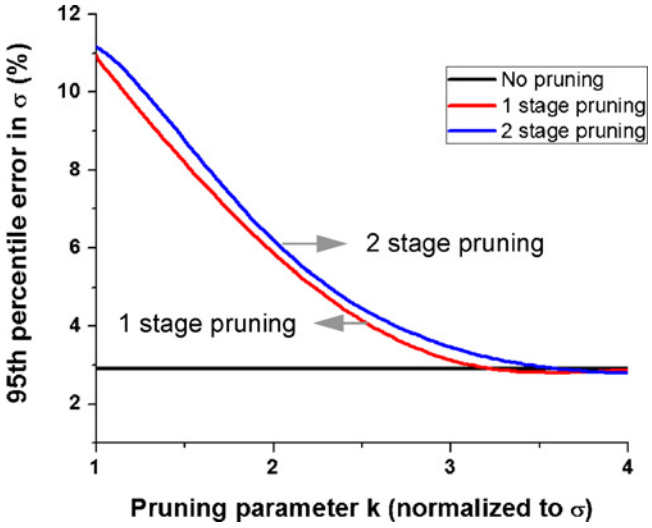

Fig. 12. Comparison of the 95th percentile error in σ for single-stage and two-stage pruning for the VD1 benchmark circuit.

TABLE VI
MEMORY USAGE AND RUNTIME PROFILE FOR SH-QMC WITH CRITICAL GRAPH IDENTIFICATION

Circuit	No. of Gates	Memory Usage (MB)	Runtime (%)		
			Sample Generation	Evaluation of Training Samples	Evaluation of Samples on Critical Graph
VD1	14 503	10.4	4.3	12.2	83.5
VD2	34 082	17.8	2 2	8.7	89.1
USB	32 898	18.3	3.1	13.9	83.0
ETHER	57 327	27.7	3.3	25.0	71.7
VGA	90 831	38.9	1.6	13.7	84.8

accuracy is obtained in all benchmark circuits evaluated. 67% of the remaining samples have error less than 0.15%.

Table VI shows the runtime profile of SH-QMC with 1-stage critical graph identification for benchmark circuits studied. The memory usage for each circuit is also shown. The sample generation step costs less than 4.3% for all the benchmark circuits. On average, 14.7% of the time is spent in the learning phase where the critical graph is identified. Note that this time does not add to the overhead (compared to when gate pruning is not performed), as the samples evaluated are a subset of the full set of SH-QMC samples; only the remaining samples require evaluation in the subsequent phase.

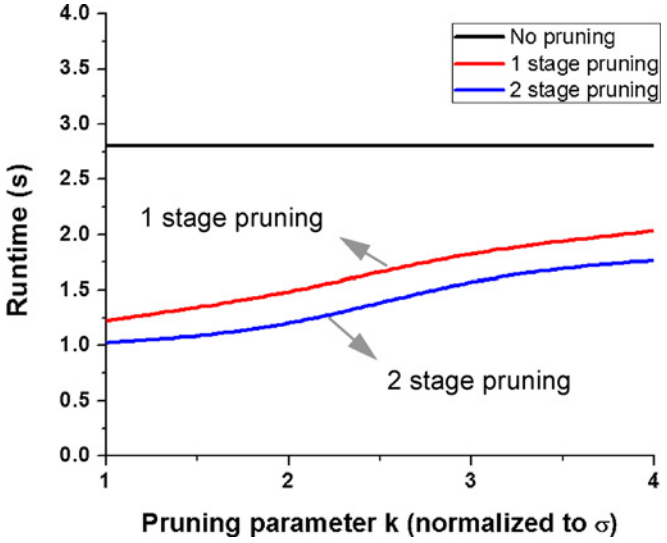

Fig. 13. Comparison of runtime versus pruning parameter for single-stage and two-stage pruning for the VD1 benchmark circuit.

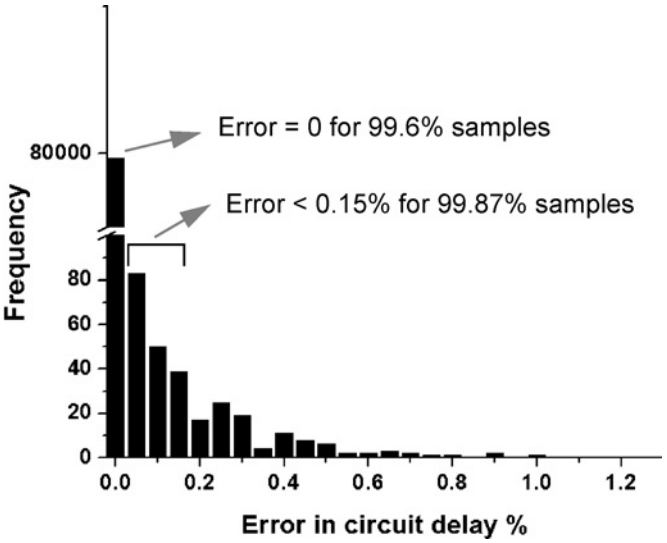

Fig. 14. Plot illustrating accuracy of the critical graph identification approach. 99.6% of the 80k total samples incur no error due to critical graph analysis relative to the non-pruned circuit STA. 67% of the remaining samples have error less than 0.15%.

C. Incremental Evaluation of Fixed Percentile Delay

Table VII presents results for the incremental evaluation of the 95th percentile and 99th percentile delay after a gate size change using our approach in Section IV. These experiments are based on process variation model A, where only channel length variation is considered. In our experiments, we selected 100 gates at random for a given circuit. Each gate was sized up individually and the percentile delays recomputed. Our simulations show that on average only 1.4% and 0.7% of samples need to be reevaluated for exact recomputation of the 95th percentile and 99th percentile delays after performing SH-QMC.

TABLE VII
PERFORMANCE OF INCREMENTAL EVALUATION OF 95TH AND 99TH PERCENTILE

Circuit	No. of Gates	Avg. Incremental Evaluations Per Gate		Avg. Incremental Evaluations Per Gate/Sample Size (%)	
		95th percentile AT	99th percentile AT	95th percentile AT	99th percentile AT
VD1	14 503	1.515	0.51	1.89	0.64
VD2	34 082	0.54	0.515	0.68	0.64
USB	32 898	1.625	0.57	2.03	0.71
ETHER	57 327	0.96	0.535	1.20	0.67
VGA	90 831	0.84	0.505	1.05	0.63

VI. CONCLUSION

This paper presented a SH-QMC approach to improve the efficiency of MC based statistical static timing analysis. The proposed approach uses easily computable timing criticality information, and requires fewer than 224 samples to achieve target accuracy with varying amounts of process variation and across all the benchmark circuits studied. With multithreading on a quad-core processor for SH-QMC, the approach is faster than traditional SSTA for comparable accuracy. Also, further speedup of SH-QMC is straightforward using parallel processing across machines. In addition, SH-QMC scales better than traditional SSTA with circuit size. We also proposed an extension to SH-QMC that employs graph pruning based on information obtained from sample evaluation. This further reduces runtime by 46–48% on average. Finally, an incremental approach to recompute a percentile delay metric after ECO is also presented. The results show that on average only 1.4% and 0.7% of original samples need to be evaluated for exact recomputation of the 95th percentile and 99th percentile delays after ECO.

The proposed SH-QMC approach has some limitations which need to be addressed in future work, in order to enable its adoption in a design optimization setting. In the proposed approach, if the designer performs a series of changes resulting in a significant change to the critical path of the design, full recomputation of samples is required. To minimize the number of recomputations, one could explore possibilities for incremental regeneration of samples and incremental critical graph analysis, with respect to change in timing criticality of the circuit graph or in the ordering of principal components.

REFERENCES

[1] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. Des. Autom. Conf.*, 2004, pp. 331–336.

[2] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *Proc. Int. Conf. Comput.-Aided Des.*, 2003, pp. 621–625.

[3] K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester, "Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation," in *Proc. Int. Conf. Comput.-Aided Des.*, 2005, pp. 1023–1028.

[4] F. N. Najm and N. Menezes, "Statistical timing analysis based on a timing yield model," in *Proc. Des. Autom. Conf.*, 2004, pp. 460–465.

[5] R. Chen, L. Zhang, V. Zolotov, C. Visweswariah, and J. Xiong, "Static timing: Back to our roots," in *Proc. ASPD*, 2008, pp. 310–315.

[6] K. R. Heloue and F. N. Najm, "Parameterized timing analysis with general delay models and arbitrary variation sources," in *Proc. Des. Autom. Conf.*, 2008, pp. 403–408.

[7] K. R. Heloue, C. V. Kashyap, and F. N. Najm, "Quantifying robustness metrics in parameterized static timing analysis," in *Proc. Int. Conf. Comput.-Aided Des.*, 2009, pp. 209–216.

[8] Y. Zhan, A. J. Strojwas, X. Li, L. T. Pileggi, D. Newmark, and M. Sharma, "Correlation-aware statistical timing analysis with non-Gaussian delay distributions," in *Proc. Des. Autom. Conf.*, 2005, pp. 77–82.

[9] L. Zhang, J. Shao, and C. C.-P. Chen, "Non-Gaussian statistical parameter modeling for SSTA with confidence interval analysis," in *Proc. Int. Symp. Phys. Des.*, 2006, pp. 33–38.

[10] S. Bhardwaj, P. Ghanta, and S. Vrudhula, "A framework for statistical timing analysis using non-linear delay and slew models," in *Proc. ICCAD*, 2006, pp. 225–230.

[11] L. Cheng, J. Xiong, and L. He, "Non-Gaussian statistical timing analysis using second-order polynomial fitting," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 1, pp. 130–140, Jan. 2009.

[12] L. Scheffer, "The count of Monte Carlo," in *Proc. TAU*, 2004.

[13] M. Keramat and R. Kielbasa, "Worst case efficiency of LHSMC yield estimator of electrical circuits," in *Proc. ISCAS*, vol. 3. 1997, pp. 1660–1663.

[14] R. Kanj, R. Joshi, and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," in *Proc. Des. Autom. Conf.*, 2006, pp. 69–72.

[15] S. Tasiran and A. Demir, "Smart Monte Carlo for yield estimation," in *Proc. TAU*, 2006.

[16] 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," in *Proc. ISQED*, 2007, pp. 685–692.

[17] A. Singhee, S. Singhal, and R. A. Rutenbar, "Practical, fast Monte Carlo statistical static timing analysis: Why and how," in *Proc. Int. Conf. Comput.-Aided Des.*, 2008, pp. 190–195.

[18] A. Singhee, S. Singhal, and R. A. Rutenbar, "Exploiting correlation kernels for efficient handling of intra-die spatial correlation, with application to statistical timing," in *Proc. Des. Autom. Test Eur.*, 2008, pp. 856–861.

[19] V. Veetil, D. Sylvester, and D. Blaauw, "Efficient Monte Carlo based incremental statistical timing analysis," in *Proc. Des. Autom. Conf.*, 2008, pp. 676–681.

[20] J. Jaffari and M. Anis, "On efficient Monte Carlo-based statistical static timing analysis of digital circuits," in *Proc. Int. Conf. Comput.-Aided Des.*, 2008, pp. 196–203.

[21] J. Jaffari and M. Anis, "Timing yield estimation of digital circuits using a control variate technique," in *Proc. ISQED*, 2009, pp. 382–387.

[22] M. Qazi, M. Tikekar, L. Dolecek, D. Shah, and A. Chandrakasan, "Loop flattening and spherical sampling: Highly efficient model reduction techniques for SRAM yield analysis," in *Proc. Des. Autom. Test Eur.*, 2010, pp. 801–806.

[23] Y. Kanoria, S. Mitra, and A. Montanari, "Loop flattening and spherical sampling: Highly efficient model reduction techniques for SRAM yield analysis," in *Proc. Des. Autom. Test Eur.*, 2010, pp. 813–818.

[24] M. Imai, T. Sato, N. Nakayama, and K. Masu, "Non-parametric statistical static timing analysis: An SSTA framework for arbitrary distribution," in *Proc. Des. Autom. Conf.*, 2008, pp. 698–701.

[25] K. Gulati and S. P. Khatri, "Accelerating statistical timing analysis using graphics processing units," in *Proc. ASP-DAC*, 2009, pp. 260–265.

[26] V. Veetil, D. Sylvester, D. Blaauw, S. Shah, and S. Rochel, "Efficient smart sampling-based full-chip leakage analysis for intra-die variation considering state dependence," in *Proc. Des. Autom. Conf.*, 2009, pp. 154–159.

[27] E. Hlawka, "Funktionen von beschrankter variation in der theorie der gleichverteilung," *Ann. Mat. Pura Appl.*, vol. 54, no. 4, pp. 325–333, 1961.
[28] R. Y. Rubinstein, *Simulation and the Monte Carlo Method*. New York: Wiley, 1981.
[29] I. M. Sobol, "The distribution of points in a cube and the approximate evaluation of integrals," *USSR Comp. Math. Math. Phys.*, vol. 7, no. 4, pp. 86–112, 1967.
[30] P. Bratley and B. Fox, "Algorithm 659: Implementing Sobol's quasirandom sequence generator," *ACM Trans. Math. Softw.*, vol. 14, no. 1, pp. 88–100, 1988.
[31] M. Stein, "Large sample properties of simulations using Latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.
[32] J. Xiong, V. Zolotov, N. Venkateswaran, and C. Visweswariah, "Criticality computation in parameterized statistical timing," in *Proc. Des. Autom. Conf.*, 2006, pp. 63–68.
[33] S. Reda and S. R. Nassif, "Analyzing the impact of process variations on parametric measurements: Novel models and applications," in *Proc. Des. Autom. Test Eur.*, 2009, pp. 375–380.
[34] M. Kanno, A. Shibuya, M. Matsumura, K. Tamura, H. Tsuno, S. Mori, Y. Fukuzaki, T. Gocho, H. Ansai, and N. Nagashima, "Empirical characteristics and extraction of overall variations for 65-nm MOSFETs and beyond," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2007, pp. 88–89.
[35] L. Pang and B. Nikolic, "Measurement and analysis of variability in 45 nm strained-Si CMOS technology," in *Proc. Custom Integr. Circuits Conf.*, 2008, pp. 129–132.
[36] ITRS. *Front End Processes* (2009, Dec. 16) [Online]. Available: http://www.itrs.net/links/2009Winter/Presentations/Conference/FEP_121609.pdf
[37] C. Clark, "The greatest of a finite set of random variables," *Oper. Res.*, vol. 9, no. 2, pp. 145–162, Mar.–Apr. 1961.
[38] A. T. Tran, D. N. Truong, and B. M. Baas, "A low-cost high-speed source-synchronous interconnection technique for GALS chip multiprocessors," in *Proc. ISCAS*, 2009, pp. 996–999.
[39] V. Veetil, Y. H. Chang, D. Sylvester, and D. Blaauw, "Efficient smart Monte Carlo based SSTA on graphics processing units with improved resource utilization," in *Proc. Des. Autom. Conf.*, 2010, pp. 793–798.
[40] F. Brglez and H. Fujiwara, "Neutral netlist of ten combinational benchmark circuits and a target translator in FORTRAN (special session on ATPG and fault simulations)," in *Proc. ISCAS*, 1985, pp. 695–698.

Vineeth Veetil received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Powai, Mumbai, India, in 2005. He received the M.S. and Ph.D. degrees, both in electrical engineering, from the University of Michigan, Ann Arbor, in 2007 and 2010, respectively.

Previously, he was a Summer Intern with the Indian Institute of Science, Bangalore, India, Lund University, Lund, Sweden, and Blaze DFM, Sunnyvale, CA, in 2003, 2004, and 2008, respectively. His doctoral research focused on developing smart Monte Carlo based algorithms for statistical static timing analysis, and their parallelization on multi-core platforms and graphics processing units. He is currently a Research and Development Engineer with Synopsys, Inc., Mountain View, CA, where he is working on the development of efficient power analysis algorithms for digital semiconductor designs.

Kaviraj Chopra received the B.E. degree in instrumentation and control from Gujarat University, Ahmedabad, India, in 2001, the M.S. degree in electrical and computer engineering from the University of Arizona, Tucson, in 2004, and the Ph.D. degree in computer science from the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, in 2008.

He was with the National Optical and Astronomical Observatory, Tucson, AZ, in 2003. He was a Summer Intern with IBM Corporation, Austin, TX, in 2005, and with Synopsys, Mountain View, CA, in 2006. He is currently a PnR Engineer with Mentor Graphics, San Jose, CA.

David Blaauw (M'00–SM'07) received the B.S. degree in physics and computer science from Duke University, Durham, NC, in 1986, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, Urbana, in 1988 and 1991, respectively.

Until August 2001, he was with Motorola, Inc., Austin, TX, as a Manager with the High-Performance Design Technology Group. Since August 2001, he has been on the faculty of the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, where he is currently a Professor. His work has focused on very large-scale integration design with particular emphasis on ultralow power and high-performance design.

Dr. Blaauw was the Technical Program Chair and the General Chair for the International Symposium on Low Power Electronic and Design, as well as the Technical Program Co-Chair and a member of the Executive Committee of the ACM/IEEE Design Automation Conference. He is currently a member of the International Solid-State Circuits Conference Technical Program Committee.

Dennis Sylvester (S'95–M'00–SM'04–F'11) received the Ph.D. degree in electrical engineering from the University of California at Berkeley, Berkeley, where his dissertation was recognized with the David J. Sakrison Memorial Prize as the most outstanding research in the UC-Berkeley Department of Electrical Engineering and Computer Science.

He is currently a Professor of electrical engineering and computer science with the University of Michigan, Ann Arbor, and the Director of the Michigan Integrated Circuits Laboratory, University of Michigan, a group of ten faculty members and more than 60 graduate students. Previously, he was a Research Staff with the Advanced Technology Group, Synopsys, Mountain View, CA, and Hewlett-Packard Laboratories, Palo Alto, CA. He was a Visiting Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He has published 300 articles along with one book and several book chapters. His current research interests include the design of millimeter-scale computing systems and energy-efficient near-threshold computing for a range of applications. He holds seven U.S. patents. He also is a Consultant and Technical Advisory Board Member for electronic design automation and semiconductor firms in these areas. He co-founded Ambiq Micro, Ann Arbor, a fabless semiconductor company developing ultralow power mixed-signal solutions for compact wireless devices.

Dr. Sylvester received the NSF CAREER Award, the Beatrice Winner Award at ISSCC, the IBM Faculty Award, the SRC Inventor Recognition Award, and eight Best Paper Awards and nominations. He is the recipient of the ACM SIGDA Outstanding New Faculty Award and the University of Michigan Henry Russel Award for distinguished scholarship. He has served on the Technical Program Committee of major design automation and circuit design conferences, the Executive Committee of the ACM/IEEE Design Automation Conference, and the Steering Committee of the ACM/IEEE International Symposium on Physical Design. He is currently an Associate Editor for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and previously served as an Associate Editor for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATED SYSTEMS. He is a member of ACM and Eta Kappa Nu.