Monte Carlo based statistical timing analysis using an efficient sampling method

Zhixi Yang, Jun Yang, Kefei Xing, and Guang Yang
College of Mechantronics Engineering and Automation, National University of Defense Technology, Changsha 410073, People’s Republic of China
a) nudhzx@163.com

Abstract: Statistical static timing analysis (SSTA) is one of the most commonly used methods for analyzing timing behavior under process variations (PVs). Monte Carlo (MC) simulation is a straightforward and effective method for SSTA. In this paper, a new sampling method is proposed to improve an MC-based approach by reducing the number of simulation samples. This method utilizes stratification with an importance analysis for the selection of samples of PVs. Extensive simulations are performed to show that the proposed method achieves a better precision on ISCAS 89 benchmarks compared to other MC simulation approaches.

Keywords: Monte Carlo, statistical static timing analysis (SSTA), efficient sampling

Classification: Integrated circuits

References
[1] D. Blaauw, et al.: “Statistical timing analysis: From basic principles to state of the art,” IEEE Trans. Comp. Aid. Design of Integr. Cir. Sys. (TCAD) 27 (2008) 589 (DOI: 10.1109/TCAD.2007.907047).
[2] A. Agarwal, et al.: “Path-based statistical timing analysis considering inter- and intra-die correlations,” Workshop on Timing Issues in Specific. Synth. of Digital Sys. (2002) 16.
[3] J. Le, et al.: “STAC: statistical timing analysis with correlation,” Proc. 41st annual Design Autom. Conf. (DAC) (2004) 343 (DOI: 10.1145/996566.996665).
[4] K. R. Heloue, et al.: “Efficient block-based parameterized timing analysis covering all potentially critical paths,” IEEE/ACM Int. Conf. Comp. Aided Design (ICCAD) (2008) 173 (DOI: 10.1109/TCAD.2008.4681570).
[5] V. Veetil, et al.: “Fast statistical static timing analysis using smart Monte Carlo techniques,” IEEE Trans. Comput. Aided Design IC Syst. 30 (2011) 852 (DOI: 10.1109/TCAD.2011.2108030).
[6] C. E. Sarnadal, et al.: Model Assisted Survey Sampling (Springer, 2003) ISBN: 0-387-40620-4.
[7] R. L. Iman: “Latin hypercube sampling,” Encyclopedia of Statistical Sciences (Wiley, Hoboken, NJ, 2006) 408.
[8] S. Bhardwaj, et al.: “A framework for statistical timing analysis using nonlinear delay and slew models,” Proc. of the IEEE/ACM Int. Conf. on Comp. Aided Des. (ICCAD) (2006) (DOI: 10.1109/ICCAD.2006.320140).
[9] M. I. Sobol: “On quasi-monte carlo integrations,” Math. Comput. Simul. 47
1 Introduction

SSTA becomes important for timing analysis under PVs [1]. Two approaches have been proposed for SSTA. The first is dubbed as the traditional SSTA [2, 3, 4]. In [2, 3, 4], delays are modeled as normal random variables (RVs). Sum and Max are the two fundamental operations to calculate the path delay when propagating the RVs to the output(s). The calculation of Max is often simplified and approximated for trading computational efficiency in [2, 3, 4]. A different approach relies on MC simulation [5]. Statistical delay distributions are collected from the statistics generated from selected samples of parameter PVs. It requires a large number of samples to attain the desired precision and speed up is met using efficient sampling techniques. [5] proposed a sampling technique which applies Stratified Sampling (SS) [6], Latin Hypercube Sampling (LHS) [7] and Quasi-Monte-Carlo-Sampling (QMC) [8, 9]. Such technique could reduce sample numbers, i.e., simulation iterations with sufficient precision.

In this paper, a novel sampling technique is proposed. Variables are grouped into important and non-important based on a ranking of variable coefficients. Both types of variables are stratified, yet according to different scenarios. Within each stratification, the expected value is chosen as a sample. Samples are then combined to form sample sets for timing analysis. Compared to previous work [5], the proposed method achieves higher accuracy at similar computational efficiency. The contributions are as following:

- Sensitivity with respect to ranking of variable coefficients is employed when analyzing the importance of the variables; stratification is then applied for both types of variable under different scenarios;
- For a stratified variable, sub-bins (or bins for non-important variable) are produced; within each bin or sub-bin, the expected value is chosen as a sample. Different samples are combined to form sample sets for timing analysis.
2 Preliminaries

2.1 Process variation model

In this paper, intra and inter-die process variations are both considered. Assume intra-die variation is spatially correlated and modeled by a multi-level quad tree partitioning [10] and each level $l$ is divided into $2^{2l}$ grids as in Fig. 1. The top level 0 has a single region while bottom level $k$ has $4^k$ grids. Assume $(k, r)$ means grid $r$ in level $k$, then the variance of $r$ is:

$$\text{Cov}(r, r) = \sigma_{\text{inter}}^2 + \sigma_{\text{intra}}^2$$  \hspace{1cm} (1)

And the covariance of $r$ and $r'$ in level $k$ of interest is:

$$\text{Cov}(r, r') = \sigma_{\text{inter}}^2 + \sum_{l=0}^{k} C_l w_l \sigma_{\text{intra}}^2$$  \hspace{1cm} (2)

where $C_l$ mean whether $r$ or $r'$ shares the same grid in the $l^{th}$ level and $C_l = 1$ means $r$ and $r'$ shares same grid, otherwise the value is 0. $w_l$ is weight of $l^{th}$ level and $\sum_{l=0}^{k} w_l = 1$. For example, in Fig. 1, the covariance of gate length ($L_g$) in bottom level $k = 2$ of region $r = 1$ (i.e. grid $(2, 1)$) with that in grid $(2, 2)$ and $(2, 16)$, i.e. when $r$ is 2 and 16 is:

$$\text{Cov}(1, 2)_{L_g} = \sigma_{\text{inter}, L_g}^2 + w_0 \sigma_{\text{intra}, L_g}^2 + w_1 \sigma_{\text{intra}, L_g}^2$$

$$\text{Cov}(1, 16)_{L_g} = \sigma_{\text{inter}, L_g}^2 + w_0 \sigma_{\text{intra}, L_g}^2$$  \hspace{1cm} (3)

Then the correlation matrix $R$ is obtained as per Eq. (1) and (2) and above models are only applicable for same source of parameter PVs. Different sources of parameter PVs are independent. Then the principal component analysis (PCA) is applied to decompose correlated parameter PVs into independent variables, making the problem tractable. Given a correlation matrix as per Eq. (1) and (2), PCA finds a set of uncorrelated and orthogonal principal components (PCs). Then, correlated PVs are expressed as a linear function of those PCs, e.g., if $L_g^s$ is the correlated gate length for grid $s$ and assume it is normally distributed with $\mu_{L_g^s}$ and $\sigma_{L_g^s}$. The correlated variation is linearly expressed by PCs as:

$$L_g^s = \left( \sum_{i=1}^{q^l} \sqrt{\lambda^s_i} \cdot v_{s,i} \cdot p_l \right) \sigma_{L_g^s} + \mu_{L_g^s}$$  \hspace{1cm} (4)
where $\lambda_s$ is $s^{th}$ eigenvalue of the correlation matrix $R$, $v_{s,i}$ is the $s^{th}$ element of $i^{th}$ eigenvector of $R$. PCs are normally distributed variables with unit variance and zero mean.

### 2.2 Process parameters and delay model

As mentioned in Section 2.1, both inter and intra die PVs are leveraged. A 100 nm technology on a two-layer interconnect metal model is used in this paper as in [11]. Table I shows the process parameters and their maximal value of deviations based on the predictions from [12, 13] for both inter and intra die variation.

Following process variation sources are considered [12], for each metal layer $l_i$, metal width $W_{int,l_i}$, metal thickness $T_{int,l_i}$ and ILD thickness $H_{ILD,l_i}$, the receiver gate width $W_g$, gate length $L_g$ and oxide thickness $T_{ox}$. In this work, both gate and interconnect delays are taken into consideration and Elmore delay model is leveraged to calculate both gate and interconnect delays. Moreover, the first order Taylor expansion which expressed as a function of process parameters of interconnect and receiver is to approximate the distributions of gate or interconnect delays, for instance, the interconnect delay could be defined as [11]:

$$d_{int} = d_{int}^0 + \sum_{s \in \varphi_{g}} \left[ \frac{\partial d}{\partial L_g^s} \right]_0 \Delta L_g^s + \sum_{s \in \varphi_{g}} \left[ \frac{\partial d}{\partial W_g^s} \right]_0 \Delta W_g^s + \sum_{s \in \varphi_{g}} \left[ \frac{\partial d}{\partial T_{ox}^s} \right]_0 \Delta T_{ox}^s $$

$$+ \sum_{n=1}^{n_{local}} \left\{ \sum_{s \in \varphi_{int}} \left[ \frac{\partial d}{\partial W_{int}^s} \right]_0 \Delta W_{int}^s + \sum_{s \in \varphi_{int}} \left[ \frac{\partial d}{\partial T_{int}^s} \right]_0 \Delta T_{int}^s + \sum_{s \in \varphi_{int}} \left[ \frac{\partial d}{\partial H_{int}^s} \right]_0 \Delta H_{int}^s \right\}$$

Since the die area is divided into grids, the interconnect tree could be with several segments in different grids; same for the gates the interconnect drives. In Eq. (5), $\varphi_g$ is set of indices of grids that all receiver gates fall in. $\varphi_{int}$ is set of indices of grids that the interconnect tree traverses. $\Delta L_g^s = L_g^s - \bar{L}_g^s$ where $L_g^s$ has the same meaning as Eq. (4). $\Delta W_g^s, \Delta T_{ox}^s, \Delta W_{int}^s, \Delta T_{int}^s, \Delta H_{int}^s$ are similarly defined. Subscript 0 next to each sensitivity represents the fact that it is evaluated at the nominal value of the delay. Thus by substituting Eq. (4) into (5), Eq. (5) could be expressed as linear function of PCs as:

$$d = d_0 + \sum_{i=1}^{M} k_i p_i$$

where $d_0$ is the nominal value of the delay, $k_i$, $i = 1, \cdots, M$ are the coefficient for the PCs. $p_i$ is independent unit normal PCs. In this paper, the total number of PCs equals to the product of grid size (i.e. $4^k$) of the bottom level in the quad tree model with the number of sources of PVs.
3 Proposed sampling method

This section presents procedures for generating sample sets using the proposed efficient sampling method (ESM). The proposed method is based on an important analysis to distinguish important from non-important PCs. A heuristic technique is used to order the variables. As per Eq. (6), totally $M$ PCs are utilized, so the first step is to rank the absolute value of PC coefficients for each gate or interconnect delay and store the indices of top $T$ coefficients as candidates in Rank (lines 1–4). Next step is to get the statistics of the values (i.e. indices of PCs) in Rank, and choose the most $t$ values as indices of important PCs (IPCs). The rest are the non-important PCs (NPCs).

Algorithm Selection of important PCs

Input: PC Coefficients of delay parameters for each gate and interconnect in Eq. (6)

Output: Indices of important PCs

1. while $i < \text{Total number of gates and interconnects}$
2. Let vector $V = \text{abs}[k_1, k_2, \cdots, k_M]$ from gate or interconnect delay $i$
3. Let $\text{Rank}[i] = \text{index of top } T$ of $V$
4. end
5. Calculate the statistics of values in Rank; choose the top $t$ values that take up most of the elements in Rank. $t$ contains the indices of IPCs. Return $t$ as important PC’s indices

Assume $p_i$ has a probability density function (pdf) as $f(p_i)$ and $\int_{-\infty}^{\infty} f(p_i)dp_i = 1$. Assume $k$ IPCs are used and each IPC $p_i$ is stratified into $m_i$ equal probable bins:

$$\int_{b_0}^{b_1} f(p_i)dp_i = \int_{b_1}^{b_2} f(p_i)dp_i = \cdots = \int_{b_{m-1}}^{b_m} f(p_i)dp_i = \int_{b_m}^{b_{m+1}} f(p_i)dp_i$$  \hspace{1cm} (7)

where $b_0 = -\infty$ and $b_m = \infty$. $b_j$s are boundaries of bins ($j = 0, 1, \cdots, m_i$ for $p_i$).

Each bin of $p_i$ must be exhaustively combined with each bin of rest $k - 1$ IPCs. Then a total size of $N = \prod_{i=1}^{k} m_i$ samples for the exhaustive list of bins of $k$ IPC combinations is obtained. Obviously, each bin must have $\frac{N}{m_i}$ samples for $p_i$. In this paper, each bin is further divided into $\frac{N}{m_i}$ equal probable sub-bins:

$$\int_{b_j}^{b_{j+1}} f(p_i)dp_i = \int_{b_{j,0}}^{b_{j,1}} f(p_i)dp_i + \int_{b_{j,1}}^{b_{j,2}} f(p_i)dp_i + \cdots + \int_{b_{j,m_i}}^{b_{j,m_i+1}} f(p_i)dp_i$$  \hspace{1cm} (8)

where $b_j = b_{j,0}$, $b_{j,1} = b_{j,0} + \frac{b_{j,m_i}}{m_i}$, $\int_{b_{j,0}}^{b_{j,1}} f(p_i)dp_i = \int_{b_{j,1}}^{b_{j,2}} f(p_i)dp_i = \cdots = \int_{b_{j,m_i}}^{b_{j,m_i+1}} f(p_i)dp_i$.

Within each sub-bin, the expected value is chosen as a sample:

$$\int_{b_{j,0}}^{b_{j,1}} p_i f(p_i)dp_i = S_{j+1,0b+1}$$  \hspace{1cm} (9)
where \( sb = 0, 1, \ldots, N - 1 \). \( S_{x,y} \) means the \( y^{th} \) sample for \( x^{th} \) bin for IPCs. For NPCs, \( N \) equal probable bins are stratified:

\[
\int_{b_0}^{b_{j+1}} f(p_i)dp_i = \int_{b_1}^{b_{j+1}} f(p_i)dp_i = \cdots = \int_{b_{N-2}}^{b_{j+1}} f(p_i)dp_i = \int_{b_{N-1}}^{b_{j+1}} f(p_i)dp_i \quad (10)
\]

where \( b_0 = -\infty \) and \( b_N = \infty \). \( b_j \) has same meaning as in Eq. (9) and \( j = 0, 1, \ldots, N - 1 \). Within each bin, the except value is chosen as a sample:

\[
\int_{b_j}^{b_{j+1}} p_i f(p_i)dp_i = S_{j+1} \quad (11)
\]

\( S_x \) means the \( x^{th} \) sample for NPCs.

A sample set is formed according to exhaustive list of bin combinations of IPCs together with randomly chosen samples from NPCs. Since each bin of IPC has multiple samples, different selection of samples lead to a different sample set.

An example illustrates the generation of samples. In Fig. 2, four PCs are used, two of which \( p_1 \) and \( p_2 \), are assumed to be IPCs while \( p_3 \) and \( p_4 \) are NPCs. Red dots are the expected values for both types of PCs as samples. Assume \( m_1 = 3 \) and \( m_2 = 2 \), then sample size is \( N = m_1 m_2 = 6 \) in the set; hence \( p_3 \) and \( p_4 \) are divided into 6 equal probable bins. Solid lines are the boundaries of bins while the dashed lines are the boundaries of sub-bins. The \( j + 1^{th} \) bin of \( p_i \) is labeled as \( B_{i,j+1} \) where \( j = 0, 1, \ldots, m_i - 1 \) for IPCs and \( j = 0, 1, \ldots, N - 1 \) for NPCs. The exhaustive list of bins of \( p_1 \) and \( p_2 \) combinations are shown in Fig. 3. As per the list of bin combinations for IPCs, within each bin, samples are chosen randomly from sub-bins. For NPCs, samples are randomly selected from bins.
4 Simulation and results

4.1 Simulation methodology

The proposed method is implemented in C++ as a software package and tested on the edge triggered sequential ISCAS89 test benchmark circuits. All experiments are tested on a Linux workstation with a 2.1 GHz CPU and 16 GB memory. Since it requires the layout information of about the location of gates and interconnects, all cells are firstly placed using the Capo tool [14]. All the nets are then routed based on placement of the cells. Due to lack of access to real wafer data, the covariance matrix for intra-die variation is from spatial correlation model discussed in Section 2 by equally splitting the variance into all levels, i.e., $w_l$ in Eq. (2) is same for any $l$ level. Depending on the size of circuit, the grid number for the spatial correlation model is varied from $2^2 / C_2^2$ to $16^2 / C_2^16$, and each grid contains less than 100 cells. To verify the results of proposed ESM, in this paper, a golden sample set with 10k randomly generated samples, referred as the golden MC, is used as reference to generate delay distribution with moments $u_{golden}$ and $\sigma_{golden}$.

To assess the proposed ESM, $G$ sample sets are leveraged, each generates a delay distribution with moments $u_i$ and $\sigma_i$, where $i = 1, 2, \cdots, G$. Then, the absolute errors of the mean and the standard deviation of the delay distribution for a sample set with respect to the golden MC, are calculated (i.e. $error_u = |u_i - u_{golden}|$ and $error_\sigma = |\sigma_i - \sigma_{golden}|$ where $i = 1, 2, \cdots, G$). Hence, statistics with $G$ absolute relative errors for mean of delay distribution is established, i.e., $R_{err,u} = \frac{error_u}{u_{golden}}$ (similarly for the standard deviation). The 95th percentile of such statistics is then used as criterion for comparing different techniques, denoted as $err_{u,95}$ and $err_{\sigma,95}$. So the smaller a value is, the closer the mean and the standard deviation are with respect to $u_{golden}$ and $\sigma_{golden}$; it also means that the samples for each sample set have high precision when finding the circuit delay.

4.2 Simulation results

In this paper, 100 (i.e. $G = 100$) sample sets are used for simulation. Top 3–5 PCs are chosen as important while the rest are the non-important.

First, an experiment is performed with varying sample size to obtain $err_{u,95}$. In fact, $err_{u,95}$ is less than 0.1% for all the simulations in this paper, indicating that the proposed method estimates mean of the delay precisely (Table II). However, the bottleneck here is to estimate $err_{\sigma,95}$ which changes greatly with varying sample
size. Fig. 4 shows \(err_{a,95}\) using ESM with different sample sizes on s38584 from benchmark set ISCAS89. Basically \(err_{a,95}\) shrinks when sample size increases, matching the fundamental facts of MC simulation. Different to normal MC, ESM doesn’t have a continuous but dispersible sample size due to stratification and permutation on important PCs. Fig. 5 is the CDF curve for golden MC and ESM with 320 samples and the CDF curve of ESM matches curve of golden MC closely, which demonstrates the precision of ESM.

[5] has proposed a Smart sampling method for MC based SSTA. It uses SS for important variables; each bin is sampled by utilizing QMC, while LHS is used for non-important variables. In this paper, ESM and Smart sampling of [5] are also compared in terms of precision and efficiency. However, Smart sampling requires
other techniques, such as variable ranking based on timing criticality and path reduction. Thus, to fairly compare the sampling self, same important and non-important variables are utilized and no path reduction is leveraged for both ESM and Smart sampling; hence only difference on the generation of the samples is to be stressed. The criterion is to find the minimal sample size of ESM to achieve $err_{\alpha,95} < 5\%$.

Table II. Minimal sample size to achieve $err_{\alpha,95} < 5\%$ for ESM

| Name | # Cells | # Grids | Minimal Sample size | $err_{\alpha,95}$ (%) | $err_{\alpha,95}$ (%) |
|------|---------|---------|---------------------|---------------------|---------------------|
| s27  | 13      | 4       | 288                 | 4.32                | 0.01                |
| s1196| 547     | 16      | 288                 | 4.82                | 0.04                |
| s5378| 2958    | 64      | 300                 | 4.35                | 0.01                |
| s9234| 5852    | 64      | 300                 | 4.66                | 0.08                |
| s13207| 8267  | 256     | 384                 | 4.85                | 0.03                |
| s15850| 10639 | 256     | 336                 | 4.52                | 0.05                |
| s38417| 23815 | 256     | 336                 | 4.76                | 0.02                |
| s38584| 20705 | 256     | 320                 | 4.75                | 0.03                |

Table II gives the benchmark circuit size and grids divided in this paper. The minimal size to attain target accuracy and $err_{\alpha,95}$ is also listed in Table II.

In order to compare the precision of ESM to Smart sampling, same sample size as in Table II is chosen for Smart sampling to calculate $err_{\alpha,95}$ and plot the $err_{\alpha,95}$ for both ESM and Smart for different circuits in Fig. 6. The $err_{\alpha,95}$ of ESM is lower than that of Smart sampling, indicating a better precision.

![Fig. 6. $err_{\alpha,95}$ for for both ESM and Smart sampling with same sample size for different circuits](image)

Table III shows the run time for both the proposed ESM and the Smart sampling approach; the run time consists of the time for generating the samples, ranking the PCs as well as the time required for the delay analysis. It is observed
that both methods have close run time with same sample size for corresponding test circuits.

Hence, the proposed sampling technique has significant improvement on accuracy compared to Smart sampling [5] with similar efficiency (i.e., run time) when assigned same number of samples.

### 5 Conclusion

In this paper, an efficient sampling technique has been proposed for statistical static timing analysis (SSTA) of VLSI circuits. An analysis and simulation based assessment indicates that the proposed method improves over existing methods with better precision with similar computational efficiency.

**Table III.** Run time for ESM and Smart sampling [5] with same sample size from Table II

| Name  | Run time (s) | Name  | Run time (s) |
|-------|--------------|-------|--------------|
|       | ESM          | Smart |              | ESM          | Smart |
| s27   | 0.01         | 0.01  | s13207       | 62.08        | 61.96 |
| s1196 | 0.5          | 0.52  | s15850       | 75.25        | 75.46 |
| s5378 | 5.31         | 5.35  | s38417       | 170.28       | 170.43 |
| s9234 | 10.75        | 10.79 | s38584       | 193.85       | 194.1 |