IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 22, 2018, accepted October 7, 2018, date of publication October 11, 2018, date of current version November 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2875474

SoftCorner: Relaxation of Corner Values for Deterministic Static Timing Analysis of VLSI Systems

HYUNJEONG KWON[1], (Student Member, IEEE), JAE HOON KIM[2], (Member, IEEE), SEOKHYEONG KANG[1], (Member, IEEE), AND YOUNG HWAN KIM[1], (Senior Member, IEEE)

[1]Department of Electrical Engineering, Pohang University of Science and Technology, Pohang 37673, South Korea
[2]Design Technology, Samsung Electronics Co., Ltd., Hwaseong 18448, South Korea

Corresponding author: Young Hwan Kim (youngk@postech.ac.kr)

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Consilience Creative Program (IITP-2018-2011-1-00783) supervised by the IITP (Institute for Information & communications Technology Promotion).

ABSTRACT We propose *SoftCorner*, which is a novel approach to estimate the worst delay (3σ delay) of a VLSI system. SoftCorner is a modification of deterministic static timing analysis to overcome its tendency to give pessimistic estimates of the system delays when it uses 3σ corner delays for logic gates. The basic idea of SoftCorner is to use corner delays that are relaxed to $<3\sigma$. To select the relaxed gate delays, SoftCorner uses the delay information obtained from the K most-critical paths and then determines the degree of relaxation, and constructs a model that represents the probability that the delay of each logic gate is selected. This probability model is used to guide the selection of gate delays; the worst system delay with relaxed delay criterion can be obtained by running the selection module until the mean of the results converges. SoftCorner can also estimate the system delay at an arbitrary percentile. In experiments, SoftCorner estimated the system delay of benchmark circuits at 99.87^{th}, 95^{th}, and 85^{th} percentiles with an average error $<2\%$ compared with the Monte-Carlo (MC) simulation, and produced the estimates 5.89×10^4 times faster than the MC simulation on average.

INDEX TERMS Statistical analysis, computer aided analysis, analysis of variance, circuit analysis, Monte-Carlo methods, deterministic static timing analysis, process variations.

I. INTRODUCTION

As the technology further scales down, fine control in the fabrication process becomes increasingly difficult, so the relative importance of process variations (PVs) has continuously increased [1]. PVs impose uncertainties on device characteristics that are affected by process parameters. As a result, timing components such as gate delay and arrival time at the circuit nodes become random variables. Therefore, designers seek to obtain limits to the system delay at a given criterion, such as three standard deviations (3σ) above the mean (i.e., 99.87 percentile), which is related to the timing yield.

Many methods to estimate system delay at the target percentile have been suggested. One approach is to estimate the form of system delay distribution, then to use a cumulative density function to determine the system delay at the target percentile. Another approach is to obtain the system delay at the target percentile directly, like the deterministic static timing analysis (DSTA).

Among the methods included in the first approach to estimate the system delay at the target percentile after obtaining the distribution, Monte-Carlo (MC) is the most fundamental and accurate. MC simulation generates random samples from the parameter spaces and uses the samples in transistor-level timing simulations [2]. After a large number of simulations, estimated system delay converges closely to the system delay. MC simulation is the most accurate method because it does not require any assumptions about the distribution of the timing components. However, MC simulation is impractically slow.

For this reason, many analytic model-based methods have been proposed that do not require numerous simulations [1], [3]–[8]. However, widespread use of these methods in industry is not practical, for several reasons. First the quantity of statistical foundry data in standardized format is insufficient [9]. Moreover, many challenges such as analyses of the latch-based design and interconnect analysis, which

TABLE 1. Characteristics of the types of SSTA [19].

	Early process-specific SSTA	Early design-specific SSTA	Late design-specific SSTA
Stage	Early	Middle	Late
Netlist	Unknown	Known	Known
Placement	Unknown	Unknown	Known
Correlations	Unknown	Unknown	Known
Advantage	- Good for optimizing the design style in early stage	- Operates on early stage (pre-placement) - Most frequently used	- Most accurate
Disadvantage	- Inaccurate	- Does not reflect the correlation	- Be used in too late stage of the design flow - Requires fully specified correlation information

have been addressed in early static timing analysis (STA) algorithms [10], [11], have not been solved [9]. Therefore, instead of analytic model-based methods, many MC-based methods have been proposed to reduce the complexity problem of the traditional MC simulation.

First, the use of various variance reduction sampling (VRS) methods to increase the convergence rate when obtaining data moments (e.g., mean and variance) has been considered as an alternative to MC simulation [12]–[14]. Veetil *et al.* [13] proposed stratification + hybrid quasi MC (SH-QMC) which uses different VRSs to different parameters considering the impacts of the parameters on the circuit delay. SH-QMC greatly reduces the number of data required to estimate the converged moments of the distributions of the timing components.

However, there are limitations to obtaining the distributions of the timing components using SH-QMC. The first limitation is related to complexity. SH-QMC greatly reduces complexity compared to MC simulation, but still requires a lot of simulations to converge compared to DSTA and analytic model-based methods. This disadvantage is often an important hurdle to meeting time-to-market constraints.

In addition, SH-QMC has disadvantages related to the accuracy. Above all, it is quite difficult to calculate the error bound or the number of samples required to converge [15]. Also, there is the loss of accuracy in estimating the system delay distribution caused by using the reduced number of data. If the system delay follows a specific density function, the parameters of the density function can be obtained using the rapidly converged moments. However, the distribution form of the system delay is not known, especially in modern process technology, although the gate delay distribution is well known to have a lognormal distribution at low voltage [16]. In this case, the system delay distribution can be estimated using a histogram. However, since the histogram is essentially a discrete function, the probability density function (PDF) information is insufficient. This drawback is exacerbated by a reduction in the number of data because the histogram is more inaccurate due to the width of the bins that expands as the number of data decreases [17], [18].

Another MC-based method was proposed in [9]. Merrett and Zwolinski [9] proposed MC static timing analysis (MCSTA) to overcome the high computational complexity of MC simulation. MCSTA first characterizes the cell delays using a number of samples extracted from PVs. Then, the numerous cell delay data are stored in variance cell libraries (VCLs). MCSTA selects and allocates one gate delay from the VCLs to each gate of the circuit. Using the selected gate delays, the traditional STA is run. This method greatly improves the efficiency compared to MC simulation, but the memory size is huge and the search time is long because it uses very large-sized libraries.

The second approach, which is the way to obtain the system delay at the target percentiles directly, involves traditional DSTA. Traditional DSTA estimates the 3σ point of the system delay distribution assuming that all gate delays have three sigma points of the gate delay distributions. However, as the with-in die (WID) variation increases in modern technology [20], the traditional DSTA is known to cause the overly pessimistic results [21]. Therefore, many DSTA-based methods have been proposed to overcome these pessimistic results. These methods are more computationally efficient than MC-based methods and are more applicable in the industry than the analytic model-based methods.

Deterministic delay models (DDMs)-based STA is a DSTA-based algorithm [22]. DDM-based STA introduces a formula for less extreme δ sigma point, $\delta < 3$ compared to the conventional 3σ corner used in DSTA. However, DDM-based STA requires some unrealistic assumptions because it uses a first-order model to derive the equation. For this reason, this method causes huge error, especially when applied to recent process technology.

We propose SoftCorner, which is a DSTA-based algorithm that reduces pessimism by using one of the candidate corner values for gate delay. SoftCorner has three advantages over the previous methods: it (1) does not require considerable simulation time as MC-based methods do; (2) requires a reasonable library size because only candidates for corner values are stored; and (3) does not require any assumptions about the distributions of the timing components or the linearity of the parameters on the timing components.

II. BACKGROUND

Statistical static timing analysis (SSTA) can be classified into three categories (Table I) according to the stage in the design flow in which it is used [19].

Early process-specific SSTA is used when the design is not specified. Therefore, no circuit information including the placement and correlation information is provided. Therefore, early process-specific SSTA is performed on generic paths. Although it this method has the advantage of allowing quick optimization of the design, the analysis is the least accurate.

Early design-specific SSTA is performed after the design is determined, but other circuit information such as placement and correlations is not known. Therefore, it can be used in the pre-placement design phase. This type of SSTA is the most practical and has moderate accuracy compared to other types of SSTA. This type of SSTA is most frequently used because it has the advantage of being enabling quick optimizations with reasonable accuracy.

Late design-specific SSTA can be performed when the netlist, placement, and correlation information have all been determined. This method has the highest accuracy, but does not allow rapid optimization of the design. Therefore, it cannot be used in situations where design decisions must be made early in the design phase [23].

SoftCorner is designed for use in early design-specific SSTA. SoftCorner can be used for quick optimization during early stages of the design flow even if placement and correlation information are lacking. In the pre-placement design step, only die-to-die (D2D) variation and uncorrelated within-die (WID) variation are determined [24]. Because the information about placement and correlation is not known at this stage, spatially-correlated WID variation is not considered. Therefore, SoftCorner requires information on the netlist, process variations including D2D and uncorrelated WID components, and the user-defined target yield.

III. PROPOSED METHOD TO REDUCE PESSIMISM OF TRADITIONAL DSTA

SoftCorner uses one of the candidate corner values stored in the library. These candidates are less extreme than the corner values that are used in traditional DSTA algorithm. By using one of these candidates, SoftCorner can reduce the pessimism of traditional DSTA method. SoftCorner consists of three steps (Fig. 1).

In Step 1, the distributions of the gate delay and output slew of all logic gates are obtained using MC simulation. Because the distributions are obtained using MC simulation, SoftCorner does not require any assumptions about the PDFs of the timing components. From the obtained distributions, several candidates of corner values are stored in the library. Each candidate has a probability of being used as a delay (or output slew) in the STA engine.

In Step 2, the probability model is determined. This probability model depends on the input netlist and the user-defined target percentile. Step 2 consists of three processes after reading the input netlist. During statistical timing analysis, the maximum path delay among K most-critical path delays is a random variable $KCP_{\max}$. Therefore, during Step 2, the upper and lower bounds of the cumulative density function (CDF) of $KCP_{\max}$ are obtained using [25] after extracting the K most-critical paths [32]. Then the target delay value is set using the obtained bounds. Finally, a probability model is adopted, which yields expected delay that is equal to the target delay; this model represents the probability that a certain a candidate will be selected as the corner value during timing analysis.

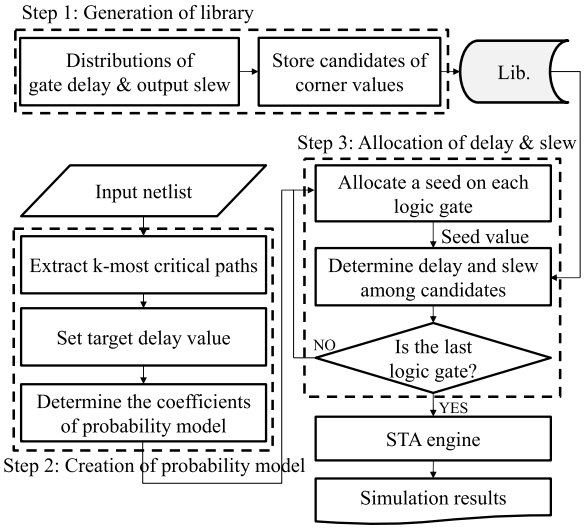

FIGURE 1. Flowchart of SoftCorner.

In Step 3, one candidate is allocated differently for each logic gate in the input circuit. For the allocation, the probability model determined in Step 2 is considered. Using the delay and slew corner values allocated to the logic gates of the circuit, the system delay at the target percentile is estimated using an STA engine. The processes of each step will be detailed in the following sections.

A. GENERATION OF LIBRARIES OF CANDIDATES FOR CORNERS

The first step of SoftCorner is generation of libraries that contain the candidate corner values. SoftCorner adopts the Liberty Variation Format (LVF), which has one delay value per load-slew combination for the logic gate [26]. Therefore, the distributions of gate delay and output slew for the i^{th} output capacitance and the j^{th} input slew are obtained using MC simulation. The candidate corner values from these distributions of MC data are stored in the libraries (e.g., Fig. 2). The traditional library only contains 3σ gate delay value for the worst case ($d_{i,j}^N$, Fig. 2) and -3σ gate delay for the best case. In contrast, SoftCorner stores N candidate corner values in the library. We set $d_{i,j}^1$ as the gate delay at the 50^{th} percentile (0σ point of the standard normal distribution) and $d_{i,j}^N$ as the gate delay at the 99.87^{th} percentile (3σ point of the standard normal distribution). These values can be changed according to the input circuits and information about process variations (Section III.B). Use of corner values $< d_{i,j}^N$ can reduce the pessimism of using only the $d_{i,j}^N$ value as traditional DSTA does. From $d_{i,j}^1$ to $d_{i,j}^N$, N candidates are stored in the library

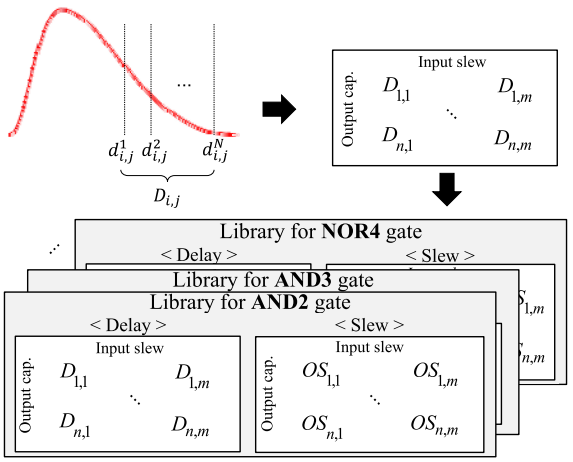

FIGURE 2. Example of a gate delay distribution at i^{th} output capacitance and j^{th} input slew.

($D_{i,j}$, Fig. 2). Similarly, the set of N candidates for the corner value of the output slew is represented as $OS_{i,j}$. For all types of logic gates, $D_{i,j}$ and $OS_{i,j}$ are stored in the library.

Because the converged MC data are used to obtain the distributions, SoftCorner does not require any assumptions about the distributions when it extracts any p^{th} percentile point ($p \leq 99.87$). One of the candidates in the library is selected and used as the corner value in STA. The candidates are less than or equal to the traditional worst gate delay ($d^N_{i,j}$). Therefore, the use of the proposed library can effectively reduce the pessimism of traditional DSTA.

B. CREATION OF PROBABILITY MODEL

The second step of SoftCorner is to construct a probability model for all the candidates of each logic gate. Each candidate $d^{i,j}_k$ has a probability P of being selected as the corner value. For each logic gate, P depends on the number of logic gates (depth) of the path in which the logic gate is included.

The expected delay of a logic gate decreases as depth of the path in which the logic gate included increases. This is because of the cancellation effect caused by the uncorrelated WID PV components. As the depth of a path increases, the uncorrelated WID PV components of the logic gates in the path are easily canceled out [27], because WID components of all logic gates in the path rarely increase or decrease concurrently, especially for deep paths.

To incorporate this cancellation effect, as the depth of a path increases, SoftCorner increases the Ps of the smaller candidates to reduce the expected value of the gate delay. In contrast, if the logic gate is included in a short path, SoftCorner decreases the Ps of the smaller candidates to increase the expected value of the gate delay. In other words, the probability model in SoftCorner follows two conditions:

(*1*) For logic gates included in a path that has large depth, P must be increased for small candidates.

(*2*) For logic gates included in a path that has small depth, P must be decreased for small candidates.

To reflect the above two conditions, the probability that the C^{th} candidate of a logic gate in a path with depth D is selected is represented as $P(C, D)$. The C value of $d^k_{i,j}$ is set such that the CDF value at $d^k_{i,j}$ (Fig. 2) is equal to that at $C\sigma$ value of a standard normal distribution. For example, $C = 3$ for $d^N_{i,j}$ because the CDF value at $d^N_{i,j}$ was set to 99.87% which is the CDF value of a standard normal distribution at the 3σ point. Including another variable x which adjusts the expected gate delay according to C and D, the probability model is represented as (1). The sum of the probabilities at all Cs is normalized to 1 by dividing $a^{C(D-x)}$ by $\sum_i a^{i(D-x)}$ because one of the N candidates must be selected.

$$P(C, D, x) = \frac{a^{C(D-x)}}{\sum_i a^{i(D-x)}} \tag{1}$$

In (1), a is a constant and x is a variable. After a is set, the value of x is determined by the input circuit and the user-defined target yield. Even if SoftCorner uses an exponential function as the probability model, other forms can be used in a similar manner, if the conditions (*1*) and (*2*) are satisfied and the existence and uniqueness of the solution of the model are guaranteed (Section IV).

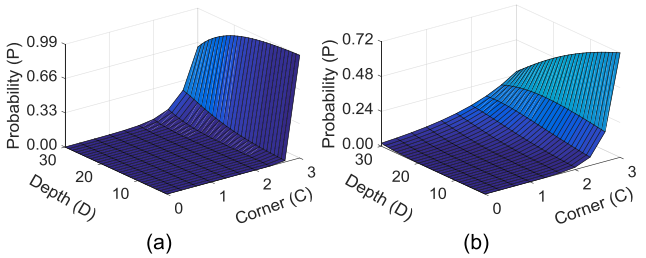

FIGURE 3. Probability model for x = 38 when (a) a = 0.7 and (b) a = 0.9.

a is related to the exponential property of the probability model. The range of a is $0 < a < 1$. The exponential property of the probability model decreases as a increases (Fig. 3a, b). Therefore, when only one logic gate is considered, a affects the variance of the selected corner values of the gate delay.

x is related to the degree of pessimism reduction. If x is too small, the smallest candidate (0σ) will be selected for all logic gates regardless of D (Fig. 4a). If x is too large, the largest candidate (3σ) will be selected for all logic gates regardless of D (Fig. 4b) and, in this case, the results of SoftCorner are the same as those of traditional DSTA. Therefore, we can reduce the pessimism by reducing x. To make SoftCorner also applicable to input circuits for which pessimism never or excessively occurs, the range of x is set to $-10 \cdot D_{\max} < x < 10 \cdot D_{\max}$.

We should determine how much to decrease x (i.e., how much to reduce the pessimism). In SoftCorner, the pessimism is reduced until the expected value of $KCP_{\max}$ using the probability model is equal to the target delay value calculated using [25]. $KCP_{\max}$ is a random variable in statistical timing analysis. Depending on process parameter values, the critical path and its delay value change every run. Therefore, $KCP_{\max}$

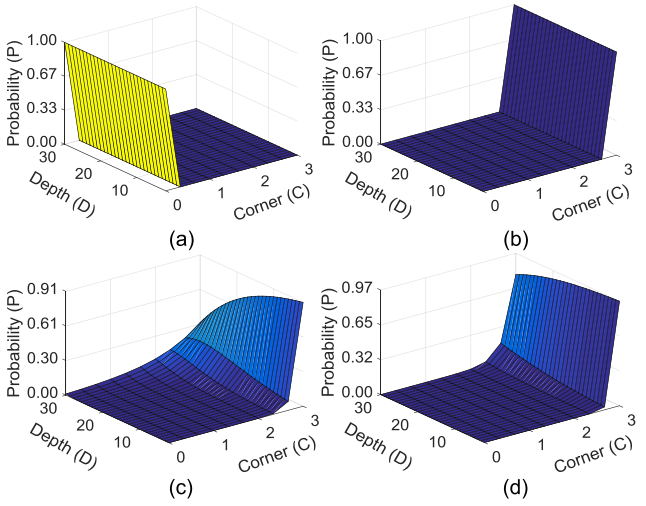

FIGURE 4. Probability model for a = 0.8 when (a) x = −300, (b) x = 300, (c) x = 35, and (d) x = 50.

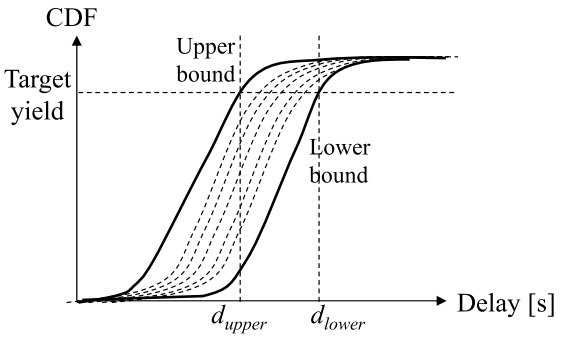

FIGURE 5. Upper and lower bounds of the CDF of KCP_{max} from [25].

is a random variable that has a certain distribution. However, the exact form of this distribution is difficult to obtain accurately even using recent methods in path-based statistical timing analysis. However, [25] proposed a method that uses stochastic majorization theory to obtain the upper and lower bounds (Fig. 5, solid lines) of the CDF of$KCP_{\max}$. In this process, the path delay is assumed to follow a normal distribution; this assumption is reasonable in most cases, because according to the central limit theorem, even though delays of each logic gate delay do not follow the normal distribution, the path delay distribution, which is the sum of several non-normal distributions, rapidly converges on a normal distribution [28]. The path delay distribution is well approximated as a normal distribution as long as $D > 4$ [16]. To validate the normality of the path delay distribution, we used the inverter chain with different depths, and made quantile-quantile (Q-Q) plot [29] and Shapiro-Wilk (S-W) test [30] which are frequently used as the normality test (Fig. 6). For each inverter, 1000 delay values are simulated using HSPICE. Q-Q plot means that if the quantiles of the samples ('+' sign in Fig. 6) are on the quantiles of the normal distribution (straight line in Fig. 6), the samples can be approximated to the normal distribution. Also, the larger the S-W test value is, the closer to normal distribution the samples are. Fig. 6 shows that the path delay distribution is close to normal distribution when the depth is larger than four.

With the reasonable preconditions, the upper d_{upper} and lower d_{lower} bounds of the CDF of $KCP_{\max}$ can be obtained (Fig. 5) even though the exact form is unknown. Among the $KCP_{\max}$ values at an ideally infinite number of process parameter value sets, the value of $KCP_{\max}$ at the target percentile is the minimum of d_{upper} and the maximum of d_{lower}.

SoftCorner first sets the target maximum path delay d_{targ}. We will represent the $KCP_{\max}$ value when the gate delays at the target yield are used as d'_{lower}. Because d'_{lower} can be considered as the worst case, we shifted d_{upper} and d_{lower} so that $d_{lower} = d'_{lower}$. Then we set

$$d_{targ} = \frac{d_{upper} + d_{lower}}{2} + |d_{lower} - d'_{lower}|. \tag{2}$$

In statistical timing analysis, d_{targ} differs from the actual target percentile system delay because one path that is not included in the K most-critical paths can be a system delay. Therefore, a path that is not included in the K most-critical paths can be the path with maximum delay. Therefore, the upper and lower bounds in [25] cannot be solely used to estimate the system delay at any percentile. SoftCorner uses the method proposed in [25] only to determine how much to reduce the pessimism. SoftCorner can consider other paths that are not included in the K most-critical paths but can be critical paths.

After setting d_{targ}, the x value is determined. This means that the probability model to be used in STA engine is fixed. x affects the Ps of the candidates and therefore affects the expected value of the gate delay value. As x increases, the P of the larger C at any D increases (Figs. 4c, d). For this reason, the expected path delay increases as x is increased. The probability model is determined so that the expected maximum path delay using the model is equal to d_{targ}.

Firstly, the expected value of $KCP_{\max}$, $\max\{\mathrm{E}[d_{\text{path},j}(x)]\}$, is calculated using a certain x. The x value when $\max\{\mathrm{E}[d_{path,j}(x)]\}$ is equal to d_{targ} is selected. In this step, the Newton-Raphson (NR) method is used as follows. The k^{th} candidate is selected considering the probability model defined in (1). Therefore, the expected delay of a logic gate included in the path of depth D is defined as in (3). If the logic gate is included in more than one path simultaneously, the minimum depth among the depths of the paths is used, to avoid optimism.

$$E[d_{gate,i}(x)] = \sum_{k} P(k, D_i, x) \cdot d_k, \tag{3}$$

where $k \in \{(n-1) \cdot \{3/(N-1)\} | n \in \mathbb{N}, 1 \le n \le N\}$ and N is the number of candidate corner values. Adding up all the $E[d_{gate,i}(x)]$ values of all the logic gates in the path, the expected path delay is calculated as

$$E[d_{path,j}(x)] = \sum_{i \in gates\ in\ j^{th}\ path} E[d_{gate,i}(x)]. \tag{4}$$

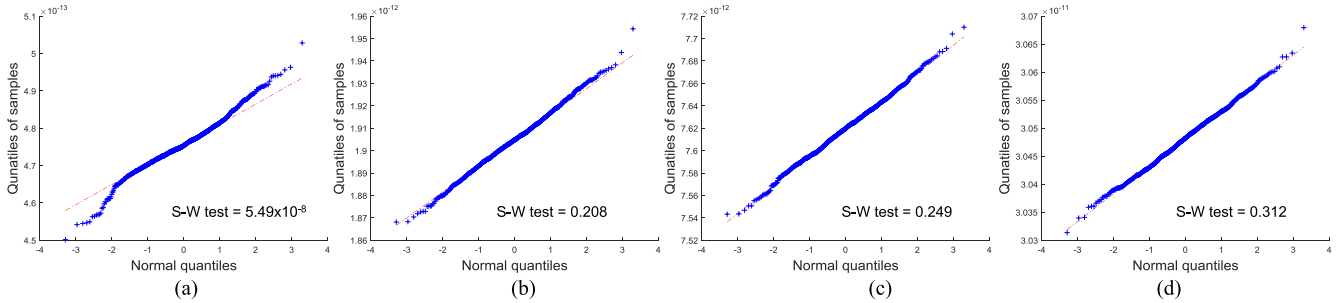

FIGURE 6. Q-Q plot and S-W test for the inverter chains when the depth is (a) 1, (b) 4, (c) 16, and (d) 64.

After calculating $E[d_{path,j}(x)]$ for all K most-critical paths, the maximum value is obtained, denoted as $\max(\mathrm{E}[d_{\mathrm{path,j}}(x)])$. The x value when $\max(\mathrm{E}[d_{path,j}(x)]) = d_{targ}$ is used in the probability model (1); i.e., the solution of

$$g(x) = \max(E[d_{path,j}(x)]) - d_{targ} = 0 \tag{5}$$

becomes x of the probability model.

The NR method yields the solution of (5). Using the NR method, x at the $(t+1)^{\mathrm{th}}$ step can be calculated using x at the t^{th} step as

$$x^{t+1} = x^t - \frac{g(x^t)}{g'(x^t)}. \tag{6}$$

where $g'(x^t)$ is the derivative of $g(x^t)$ with respect to x, as

$$\begin{aligned} g(x^t) &= \max\left(\sum\nolimits_i E[d_{gate,i}(x^t)]\right) - d_{targ} \\ g'(x^t) &= \max\left(\sum\nolimits_i E'[d_{gate,i}(x^t)]\right) \end{aligned} \tag{7}$$

where $E[d_{gate,i}(x^t)] = \sum_{k=1}^{N} P(k, D_i, x^t) \cdot d_k$, $E'[d_{gate,i}(x^t)] = \sum_{k=1}^{N} P'(k, D_i, x^t) \cdot d_k$ and $P'(k, D_i, x^t)$ is the derivative of $P(k, D_i, x^t)$ with respect to x as

$$P'(C, D, x) = \frac{-r(x) \ln a \cdot \{p(x) + q(x)\}}{t^2(x)}. \tag{8}$$

where $r(x) = a^{C(D-x)}$, $t(x) = \sum_i a^{i(D-x)}$, $p(x) = C \cdot t(x)$, and $q(x) = \sum_i (-i) \cdot a^{i(D-x)}$. The existence and uniqueness of the solution of (5) will be proven in section IV.

We now show the range in which the result of SoftCorner is located. Using (1), $KCP_{\max}$ is expected to be equal to d_{targ}, because the solution of (5) is chosen as x. Even though SoftCorner uses the probability model that makes the expected $KCP_{\max}$ equal to d_{targ} to determine the degree by which pessimism is reduced, the probability model is applied globally to all logic gates in the circuit in the next step. Therefore, the method can consider paths that are not included in the K most-critical paths but that can be the most critical path. As a result, SoftCorner usually gives $KCP_{\max}$ which is larger than d_{targ}. In addition, because each logic gate has relaxed candidates less than 3σ point, the result of SoftCorner is smaller than that of the traditional DSTA. Therefore, SoftCorner method is more effective than both traditional DSTA and [25].

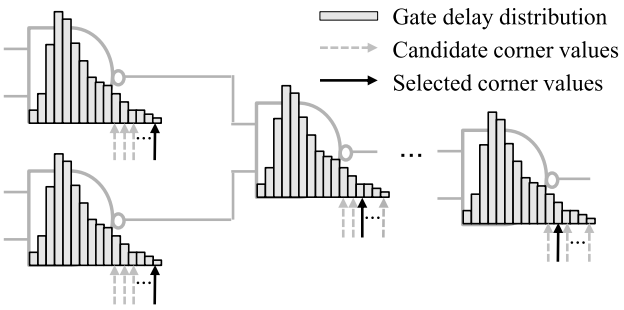

FIGURE 7. A simple example circuit and the allocated gate delay values (solid arrows) as corner values among the candidate corner values (dotted arrows).

C. ALLOCATION OF DELAY AND SLEW VALUES

One of the candidates in the library is allocated to each logic gate in the circuit (Fig. 7). The allocation is performed considering the probability $\mathrm{P}(k, D, x)$ that the k^{th} corner will be selected (III.B). Various ways to consider the probability $\mathrm{P}(k, D, x)$ may be available in the allocation step. SoftCorner considers the probability using a seed value that is extracted from the standard normal distribution.

The process of allocation step is as follows.

(i) We divide the range of a random variable that follows the standard normal distribution so that each range corresponds to a specific candidate.

(ii) For a logic gate, a random seed is generated from the standard normal distribution.

(iii) According to the range to which the random seed belongs, the corresponding candidate is selected and used as the corner value for the logic gate.

In step (i), the range is divided so that both probabilities that the seed is generated and the corresponding candidate is selected should be same. Therefore, in the first step, a random variable that follows the standard normal distribution is divided into N range $(R_k, 1 \le k \le N)$ as

$$\begin{aligned} P(1, D, x) &= f(R_1) = f(d < r_1) \\ P(2, D, x) &= f(R_2) = f(r_1 < d < r_2) \\ &\vdots \\ P(N, D, x) &= f(R_N) = f(r_{N-1} < d), \end{aligned} \tag{9}$$

so that the CDF of the k^{th} section, $f(R_k)$, is equal to $P(k, D, x)$ in (1), where $f(x)$ is PDF of the standard normal distribution. Because the $P(k, D, x)$ is determined in the previous section III.B, r_k $(1 \leq k \leq N-1)$ can be set satisfying (9).

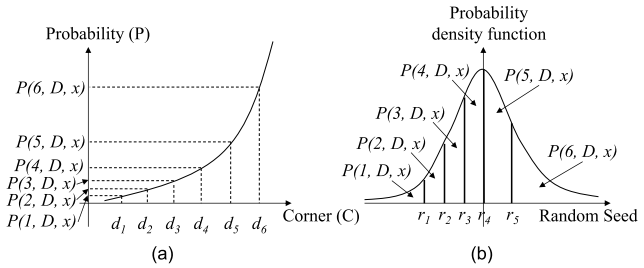

FIGURE 8. (a) The probability of candidates of a logic gate and (b) the standard normal distribution from which a random seed comes and the corresponding candidate of each range.

In Fig. 8, we provide an example for $N = 6$ case; one range corresponds to one candidate. In this example, if the generated random seed d is in $r_1 < d < r_2$, the second candidate is selected because the probability is the same with the probability of the second candidate, $P(2, D, k_3)$.

In this manner, the allocation module can simply follow the probability model determined in III.B. Using the allocated candidate values, the STA engine is run, and produces the system delay at the target yield.

IV. EXISTENCE AND UNIQUENESS OF THE SOLUTION OF PROBABILITY MODEL

A. EXISTENCE

The existence of the solution of (5) can be proved as follows. $\mathrm{E}[d_{path,j}(x)]$ is an increasing function of x. Therefore, a solution of (5) exists when the sign of $g(x)$ changes from negative to positive as x increases (Fig. 9a). To verify that (5) satisfies the condition, we should examine the ranges of d_{targ} and $\max(\mathrm{E}[d_{path,j}])$ terms in (5).

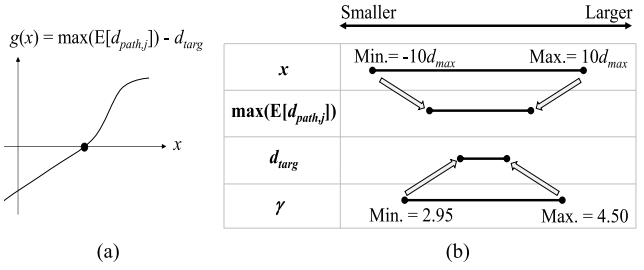

FIGURE 9. (a) The graph of g(x) and (b) the conditions of the terms in g(x), when the solution of (5) exists.

First, d_{targ} is determined as the maximum value among $\gamma \cdot \sigma_i + \mu_i$ values of K most-critical paths where μ_i and σ_i are the mean and standard deviation of the i^{th} critical path delay [25]. γ depends on K and the correlations among the K most-critical paths. From our experiments based on [25], γ ranged from 2.95 to 4.5, when $10 \leq K \leq 300$. In Fig. 9(b), the range of γ and the corresponding range of d_{targ} are described.

Next, $\max(\mathrm{E}[d_{\mathrm{path,j}}(x)])$ increases as x increases (Fig. 4). When the minimum value of $x = -10d_{max}$ is used, the result is the same as $KCP_{\max}$ when the minimum candidate in the library is used for all logic gates (Fig. 4a). Similarly, when the maximum value of $x = 10d_{max}$ is used, $\max(\mathrm{E}[d_{\mathrm{path,j}}(x)]) = KCP_{\max}$ when the maximum candidate is used for all logic gates (Fig. 4b). The solution of (5) exists when d_{targ} is between the $\max(\mathrm{E}[d_{path,j}(x)])$ value at the minimum x and the $\max(\mathrm{E}[d_{path,j}(x)])$ value at the maximum x; i.e., for a solution to exist, the minimum value of d_{targ} must be larger than the minimum value of $\max(\mathrm{E}[d_{path,j}])$ and the maximum value of d_{targ} must be smaller than the maximum value of $\max(\mathrm{E}[d_{path,j}(x)])$ (Fig. 9b). Therefore, if the circuit satisfies both of the following conditions, a solution to (5) exists.

(*3*) $\max(\mathrm{E}[d_{path,j}(x)])_{\min} < d_{targ,min}$, where $\max(\mathrm{E}[d_{path,j}(x)])_{\min} = KCP_{\max}$ when the smallest candidate $(d^1_{i,j})$ is used for all logic gates and $d_{targ,min} = d_{targ}$ when the minimum γ value is used.

(*4*) $\max(\mathrm{E}[d_{path,j}(x)])_{\max} > d_{targ,max}$, where $\max(\mathrm{E}[d_{path,j}(x)])_{\max} = KCP_{\max}$ when the largest candidate $(d^N_{i,j})$ is used for all logic gates and $d_{targ,max} = d_{targ}$ when the maximum γ value is used.

Based on *Conditions* (*3*) and (*4*), the existence of the solution of (5) can be simply tested before determining the probability model. Also, the existence of the solution can be easily satisfied by changing the range of the candidates in the library. If the pessimism of the circuit is too severe to satisfy *Condition* (*3*), the value $d^1_{i,j}$ of the first candidate should be decreased. In the same manner, in rare cases, the value of $d^N_{i,j}$ should be increased if *Condition* (*4*) is not satisfied due to weak pessimism. The above two modifications to satisfy the existence conditions can be performed without too much extra burden of runtime.

B. UNIQUENESS

To demonstrate the uniqueness of the solution of the probability model, we should prove that (5) is a strictly monotonic increasing function. That is, we should prove that $g'(x) > 0$ for all x. If (3) is a strictly monotonic increasing function, then (5) is also a strictly monotonic increasing function. Therefore, we will show the uniqueness of the solution by verifying that (3) is a strictly monotonic increasing function. Firstly, the derivative of (3) is

$$\begin{aligned}\frac{dE[d_{gate,i}(x)]}{dx} &= \sum_k \frac{-\ln(a)\sum_i (k-i)a^{(k+i)(D-x)}}{\left(\sum_i a^{i(D-x)}\right)^2} d_k \\ &= \frac{-\ln(a)}{\left(\sum_i a^{i(D-x)}\right)^2} \sum_k \sum_i (k-i)a^{(k+i)(D-x)} d_k \qquad (10)\end{aligned}$$

where $k, i \in \{(n-1)\cdot\{3/(N-1)\} | n \in \mathbb{N}, 1 \leq n \leq N\}$. Then to prove that $E[d_{gate,i}(x)]$ is a strictly monotonic increasing function, we should show that $dE[d_{gate,i}(x)]/dx > 0$ for all x.

In (10), $-\ln(a)/\left(\sum_i a^{i(D-x)}\right)^2 > 0$ is always true because $0 < a < 1$. Therefore, (11) should be proven to show that (5) is a strictly monotonic increasing function.

$$\sum_k \sum_i (k-i)a^{(k+i)(D-x)}d_k > 0 \tag{11}$$

The sign of each term, $(k-i)a^{(k+i)(D-x)}d_k$, in (11) can be positive or negative depending on the relative size of k and i because $a^{(k+i)(D-x)}d_k$ is always positive.

Now, we will prove (11) by showing that the sum of two terms of (11) when $(k, i) = (m, n)$ and $(k, i) = (n, m)$ where m and n are different arbitrary elements from the set from which k and i are originated is always larger than or equal to zero as in (12).

$$(m-n)a^{(m+n)(D-x)}d_m + (n-m)a^{(n+m)(D-x)}d_n > 0 \tag{12}$$

Because (11) is the summation of $(k-i)a^{(k+i)(D-x)}d_k$ for all combinations of k and i which are from the same set, proving (12) for all arbitrary m and n from the set can be the direct proof of (11).

First, we assume that $m > n$, but the proof using the opposite assumption is also possible in the same manner. Because $m > n$, the first and second terms of the left hand side of (12) are positive and negative, respectively. Meanwhile, $d_m > d_n$ if $m > n$ because we stored the candidates in the library that way as described in Section III.A. Therefore, the absolute value of the first term is always larger than that of the second term as in (13). Therefore, (12) is always true because the absolute value of the positive term is always larger than that of the negative term.

$$\left|(m-n)a^{(m+n)(D-x)}d_m\right| > \left|(m-n)a^{(m+n)(D-x)}d_n\right| \ (\because d_m > d_n)$$
$$\rightarrow \left|(m-n)a^{(m+n)(D-x)}d_m\right| > \left|(n-m)a^{(m+n)(D-x)}d_n\right| \tag{13}$$

Because (12) is proved as described above, (11) is also always true. Now, we prove that $dE[d_{gate,i}(x)]/dx$ in (10) is always positive. Therefore, (5) has a unique solution.

V. APPLICATION OF PROPOSED METHOD IN DESIGN FLOW

SoftCorner can be used in the flow of early design-specific SSTA. SoftCorner belongs to the early design-specific SSTA methods in pre-layout verification (Fig. 10). The proposed module has the netlist, the information of PV, and the target yield as inputs. In addition, a timing constraint d_{const} is given. Running the proposed module yields the system delay d_{sys} at the target percentile. The meaning of d_{sys} is that the system delay must be d_{sys} to achieve the target yield. Therefore, the design is passed if $d_{const} \geq d_{sys}$. Otherwise, the yield at the timing constraint is lower than the target yield. For that case, the design should be modified to achieve the target yield.

A user can use SoftCorner to determine the yield of the design at the pre-defined d_{const}. First, the user enters the expected yield as the user-defined target yield input. If $d_{sys} = d_{const}$, the expected yield is the yield at d_{const}. If $d_{sys} < d_{const}$, the yield at d_{const} is larger than the expected yield. In this case, the yield at d_{const} can be obtained by re-running SoftCorner with the increased expected yield until the condition $d_{sys} = d_{const}$ is satisfied. Otherwise, if $d_{sys} > d_{const}$, the yield at d_{const} is smaller than the expected yield. Therefore, the yield at d_{const} can be obtained by re-running SoftCorner with the decreased expected yield until the condition $d_{sys} = d_{const}$ is satisfied.

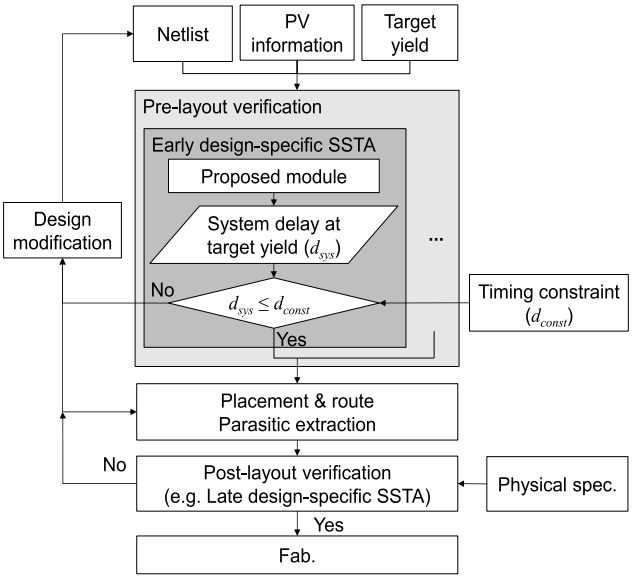

FIGURE 10. Application of SoftCorner in the design flow.

VI. EXPERIMENTAL RESULTS

A. EXPERIMENTAL ENVIRONMENT

In this section, SoftCorner was compared with the various benchmark methods. Two MC-based methods and two DSTA-based methods were used as the benchmark methods. SH-QMC [13] and MCSTA [9] were used as MC-based benchmark methods. These methods first estimate the distribution of the system delay and obtain the system delay at the target percentiles. Therefore, we ran these methods including MC simulation until from the first to the third moments of the distribution converge. Traditional DSTA and DDM-based STA [22] were used as the DSTA-based benchmark methods. Also, SoftCorner directly estimates the values at the target percentiles, so we ran it until the mean values of the simulation results converged. We compared the errors and runtime that occur when estimating the system delays at the several target percentiles. The errors were evaluated by comparing the results with the converged MC simulation results.

In experiments, 7-nm predictive technology model-multi gate (PTM-MG) [31] was used as the transistor model. The variations of the gate length, oxide thickness, fin thickness, and fin height were considered. For D2D variations, the 3σ values of the process parameter distributions were set to 18% of their mean values. For random WID variations, the

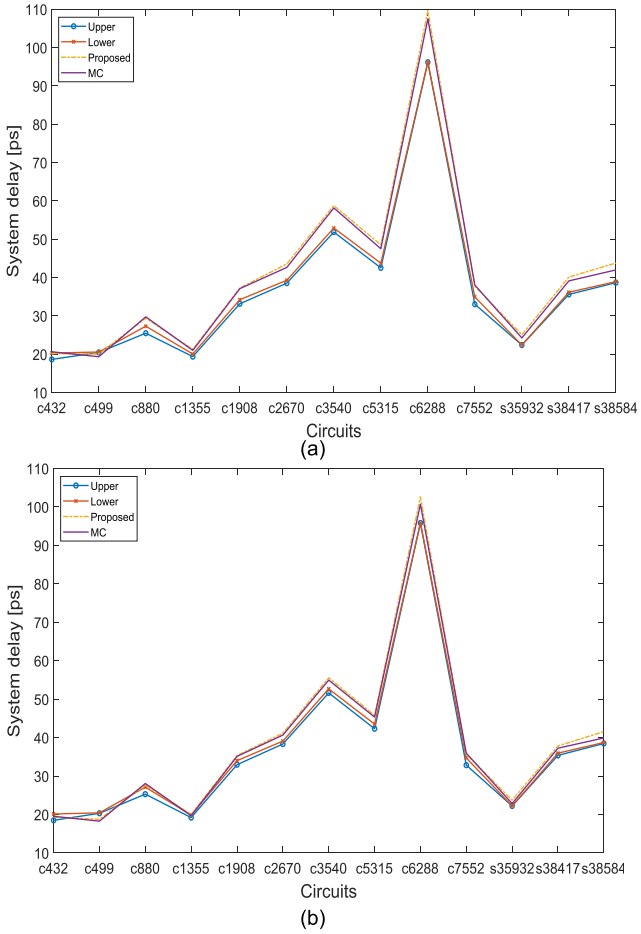

FIGURE 11. Upper and lower bounds of KCP_{max} at the (a) 99.87th and (b) 90th percentiles which are lower than the real percentile point of system delay (MC simulation results).

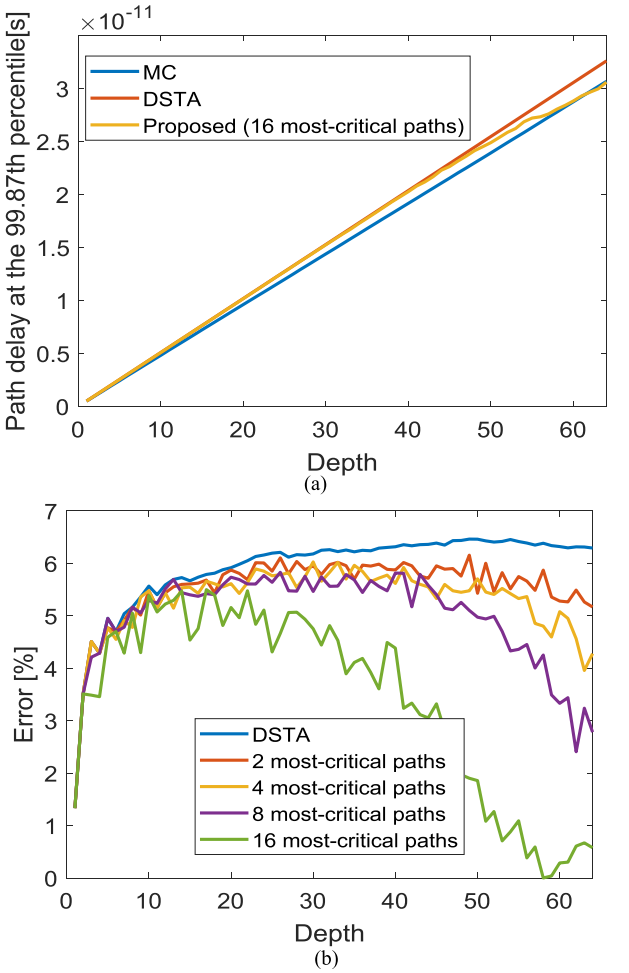

FIGURE 12. (a) Estimated path delay at the 99.87th percentile with different depths using DSTA and the proposed method, and (b) their percentage error compared to MC simulation. The proposed method uses different K most-critical paths to compare the errors according to K values.

3σ values were set to 6% of their mean values. For traditional DSTA, the 3σ point of the gate delay distribution was used as the gate delay value to estimate the 3σ point of the system delay distribution. In SH-QMC, the parameter s was set to 2.5; this is the standard for selecting the critical parameters. Among the four process parameters, two were assumed to be critical, one was assumed to be moderately critical, and one was assumed to be non-critical. For MCSTA, 10,000 instances for each logic gate were included in VCL. For SoftCorner, a of the probability model was set to 0.7 and can be changed as long as the existence of the solution is guaranteed. Also, with reference to [25], 300 paths were extracted to obtain the bounds of $KCP_{\max}$. The experiments were performed on ten ISCAS 85 and three ISCAS 89 benchmark circuits. We implemented SoftCorner and the benchmark methods in C++ language on Intel$^{(R)}$ Xeon (R) E5-2690 @ 2.90 GHz.

B. ACCURACY COMPARISON

SoftCorner uses the upper and lower bounds of $KCP_{\max}$ in Step 2 to determine the probability model. MC simulation results were generally larger than the upper and lower bounds (d_{upper} and d_{lower} in Section III.B) obtained from [25] (Figs. 11a, b). This difference is due to paths that are not included in the K most-critical paths but can have maximum path delay in some PV samples. SoftCorner successfully considered those paths because it just uses the bounds to determine the degree of pessimism reductions, then applies that probability model to the whole circuit.

The variable of the probability model is determined by K most-critical paths of the input circuit. Therefore, if the probability model is applied to the whole circuit, it can result in the error in estimating the path delay especially for the paths which are not included in the K most-critical paths. However, applying the probability model, which is determined based on the K most-critical paths, to the whole circuit does not affect critical error in estimating the system delay. To validate that, we estimated the path delay at 99.87 percentile using DSTA and the proposed method compared to MC simulation (Fig. 12). We first made a circuit which consists of several inverter chain with different depth from 1 to 64. Then, we found the probability model following SoftCorner

TABLE 2. Comparison of accuracy of estimating the system delay at the 99.87th percentile.

Circuit	Methods										
	MC delay [ps]	MCSTA		SH-QMC		DSTA		DDM-based STA		Proposed	
		Delay [ps]	Error [%]	Delay [ps]	Error [%]	Delay [ps]	Error [%]	Delay [ps]	Error [%]	Delay [ps]	Error [%]
c432	20.64	19.31	-6.47	20.32	-1.58	23.58	14.25	24.81	20.18	20.18	-2.25
c499	19.35	18.19	-6.01	19.21	-0.71	21.88	13.09	27.58	42.54	20.04	3.57
c880	29.76	27.72	-6.86	29.75	-0.03	33.80	13.58	32.27	8.42	29.41	-1.19
c1355	21.01	19.57	-6.85	21.32	1.47	23.56	12.14	22.26	5.95	21.28	1.30
c1908	37.11	34.65	-6.63	36.19	-2.50	42.67	14.96	41.14	10.84	37.27	0.41
c2670	42.66	40.13	-5.91	42.41	-0.59	49.28	15.53	47.60	11.58	43.58	2.15
c3540	58.18	54.05	-7.10	58.67	0.85	66.39	14.11	63.74	9.57	58.80	1.07
c5315	47.53	44.65	-6.05	47.60	0.15	54.63	14.93	52.28	10.00	48.54	2.13
c6288	107.49	98.85	-8.04	105.95	-1.43	119.49	11.17	108.91	1.32	109.69	2.05
c7552	38.05	35.41	-6.92	37.66	-1.01	43.50	14.33	41.75	9.72	37.78	-0.71
s35932	24.28	22.75	-6.30	23.70	-2.41	27.71	14.09	26.58	9.44	25.08	3.27
s38417	39.18	36.76	-6.18	38.45	-1.87	45.13	15.20	43.23	10.34	40.14	2.45
s38584	42.11	39.42	-6.40	41.15	-2.28	48.94	16.20	47.49	12.76	43.80	3.99
Avg. |error|	-	-	6.60	-	1.30	-	14.12	-	12.51	-	2.04

algorithm using different K values. The probability model was applied to every logic gate, and the every path delay was estimated at the 99.87th percentile (Fig. 12a). Compared to MC simulation results, we have plotted the path delay error in Fig. 12b. The error of estimated path delay decreases as K value increases. However, the proposed method shows much smaller error compared to DSTA for all path depths and for all K values. Even though the error is large especially for the short path, the error of long path delay is much important to estimate the system delay compared to that of short path because long path delays are more likely to be the system delay.

The data on y-axis corresponding to "MC" and "Proposed" legend of Fig. 11 are listed on the "MC" and "Proposed" columns of Table II, respectively, to validate the accuracy of the proposed method compared with other methods. Estimated system delays at the 99.87th percentile were compared for the proposed method and the benchmark methods ('Delay' column in Table II). Also, the percentage errors of the estimated system delays compared to MC simulation are shown ('Error' column in Table II). Mean absolute percentage errors are shown in the last row of Table II. Among the MC-based algorithms, SH-QMC estimated the target system delay most accurately (1.30% mean absolute percentage error). MCSTA selects random delay values from VCLs in gate level. Therefore, the probability that all the logic gates simultaneously selects the extreme values of the logic gate delay distribution among 10,000 instances in VCL is very low. Therefore, estimated system delay was ∼6.6% lower than MC simulation results. Among the DSTA-based algorithms, SoftCorner showed the most accurate results (2.04% average error). DSTA showed overly pessimistic results as expected because it assumes that all logic gates have

TABLE 3. Comparison of accuracy of estimating the system delay at the 95th and 85th percentiles.

Ckt.	95th percentile			85th percentile0		
	MC	Prop.	Err.	MC	Prop.	Err.
c432	19.72	19.33	-1.98	19.43	18.77	-3.42
c499	18.46	18.76	1.67	18.15	18.40	1.38
c880	28.35	27.65	-2.48	27.90	27.15	-2.69
c1355	19.90	20.03	0.69	19.59	19.65	0.33
c1908	35.42	35.34	-0.25	34.95	34.82	-0.35
c2670	41.05	41.14	0.22	40.41	40.39	-0.05
c3540	55.47	55.57	0.18	54.61	54.64	0.04
c5315	45.74	45.80	0.14	45.06	44.99	-0.14
c6288	101.7	102.7	0.97	100.1	100.9	0.79
c7552	36.28	35.69	-1.61	35.77	35.07	-1.95
s35932	23.05	23.75	3.05	22.72	23.55	3.67
s38417	37.61	37.84	0.61	37.05	37.26	0.57
s38584	40.28	41.57	3.22	39.69	41.26	3.98
Avg. |error|	-	-	1.31	-	-	1.49

3σ values of gate delay distributions. DDM-based STA uses $\delta < 3$ to use $\delta\sigma$ points of gate delay when estimating the 3σ value of system delay distribution. However, this method has limitations in accuracy especially when the recent technology is used because the method is based on the first-order model and uses several unrealistic assumptions. The accuracy of SoftCorner was comparable with that of SH-QMC which is the most accurate MC-based method. SoftCorner can estimate other percentile points with high accuracy. When the target percentiles were 90th and 85th, SoftCorner estimated the system delays percentiles with 1.31% and 1.49%

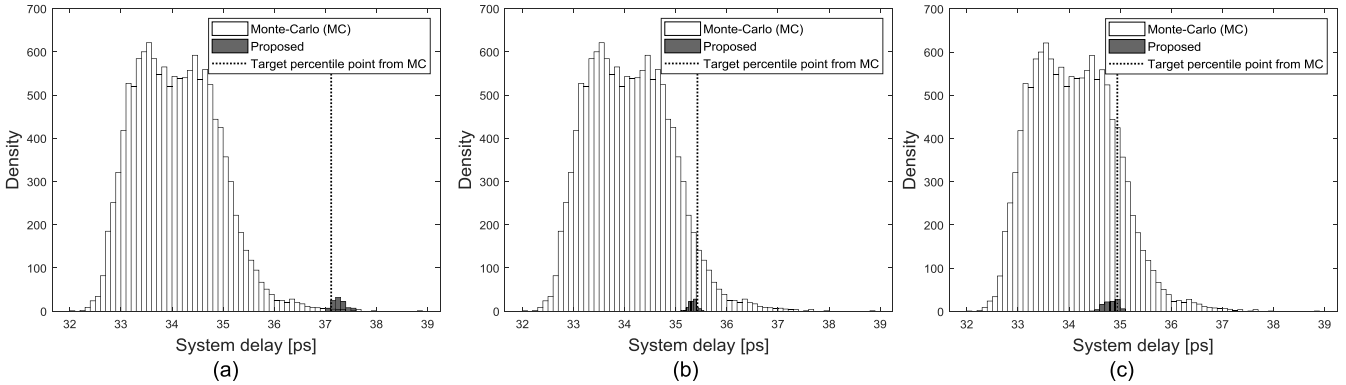

FIGURE 13. Accuracy and variance of SoftCorner (Dotted line: system delay at target percentiles, black bar: histogram of the results of SoftCorner, white bar: histogram of the system delay distribution). Circuit: c1908 / target percentiles: (a) 99.87%, (b) 95%, and (c) 85%.

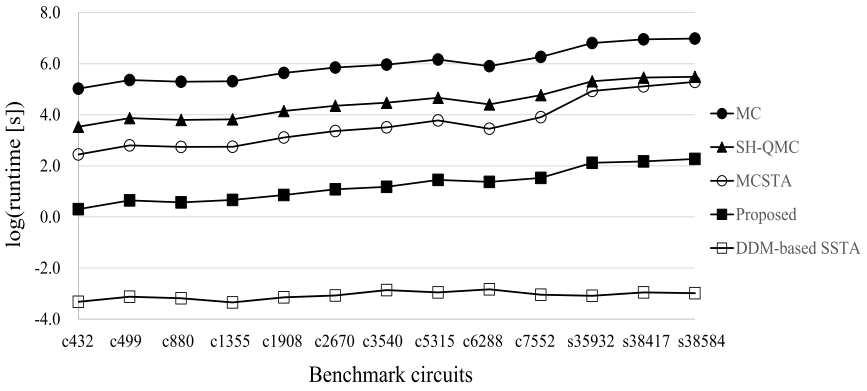

FIGURE 14. Runtime comparison.

mean absolute percentage errors compared to MC simulation results (Table III).

SoftCorner has variations on estimation results. The reason that the variation occurs is that SoftCorner determines the probability model, and a certain selected corner value changes at every run. However, because the expected value of the maximum among the K most-critical paths is set as d_{targ}, the variation is not large and not critical. The histogram of the results of SoftCorner and MC simulation to present the mean and standard deviation of the results of SoftCorner overlap (Fig. 13). When the c1908 circuit was used and the target percentile was 99.87, the mean of the results of SoftCorner was 37.27 ps with 0.41% error compared to the MC simulation results (Fig. 13, dotted line). The standard deviation (s.d.) of the results of SoftCorner was 0.12 ps which is the much smaller value than its mean value. Similarly, when the target percentiles were 95 and 85, the mean values of the results of SoftCorner were 35.42 ps (s.d. $=$ 0.08 ps) and 34.95 ps (s.d. $=$ 0.13 ps), respectively. The mean and standard deviation of the MC results were 34.41 ps and 0.84 ps, respectively. Therefore, the proposed method converges much faster than MC simulation because of the smaller standard deviation of the results from SoftCorner. In other words, SoftCorner provides the estimation results at the target percentile much faster than MC simulation.

C. RUNTIME COMPARISON

All methods reduced runtimes compared to that of MC (Fig. 14). Firstly, MCSTA reduced the runtime to 1.37×10^{-2} times that of MC simulation on average; this reduction occurs because MCSTA replaces the HSPICE timing model with the VCL. It was efficient to reduce the runtime for one simulation at the expense of accuracy. DDM-based STA greatly reduced the runtime to 3.67×10^{-10} times, compared to MC simulation, but DDM-based STA uses impractical assumptions, so it is not applicable especially in modern technology. SH-QMC greatly reduced the number of simulations, but it still required much longer runtime compared to other benchmark methods; the reason is that SH-QMC is an MC-based algorithm and does not reduce the runtime for one simulation. SoftCorner reduced runtime to 5.90×10^{-4} times that of SH-QMC and to 1.89×10^{-5} times that of MC simulation.

VII. CONCLUSION

This paper has presented SoftCorner, which is a novel DSTA-based statistical timing analysis method that uses less-extreme corner values than existing methods. SoftCorner can be used for the users who want to estimate the system delay at an arbitrary target percentile. SoftCorner first uses MC simulation data to develop libraries that contain the candidate corner values. Therefore, SoftCorner does not require any

assumptions about the distribution forms of the timing components. Depending on the input circuit and the target yield, SoftCorner constructs the probability model and uses it to guide allocation of different corner values for different logic gates. Experiments verified the effectiveness of SoftCorner by comparing the accuracy and the runtime of SoftCorner with other benchmark methods. SoftCorner estimated the system delays at 99.87^{th}, 95^{th}, and 85^{th} percentiles with average errors of 2.04%, 1.31%, and 1.49% respectively with respect to MC simulation results. This was comparable to the most accurate MC-based benchmark method, SH-QMC. However, because SH-QMC is an MC-based method, it required much longer runtime than SoftCorner. SoftCorner consumed only 5.90×10^{-4} times as much runtime as SH-QMC.

REFERENCES

[1] S. Ramprasath, M. Vijaykumar, and V. Vasudevan, "A skew-normal canonical model for statistical static timing analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 24, pp. 2359–2368, Jun. 2016.
[2] A. Srivastava, D. Sylvester, and D. Blaauw, "Statistical models and techniques," in *Statistical Analysis and Optimization for VLSI: Timing and Power*, 1st ed. New York, NY, USA: Springer, 2005, pp. 14–25.
[3] C. Visweswariah *et al.*, "First-order incremental block-based statistical timing analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2170–2180, Oct. 2006.
[4] P. J. Bhaghath and S. R. Ramesh, "A survey of SSTA techniques with focus on accuracy and speed," *Int. J. Comput. Appl.*, vol. 89, no. 7, pp. 21–25, 2014.
[5] S. Vrudhula, J. M. Wang, and P. Ghanta, "Hermite polynomial based interconnect analysis in the presence of process variations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2001–2011, Oct. 2006.
[6] L. Zhang, J. Tsai, W. Chen, Y. Hu, and C. C.-P. Chen, "Convergence-provable statistical timing analysis with level-sensitive latches and feedback loops," in *Proc. ASP-DAC*, Yokohama, Japan, 2006, pp. 941–946.
[7] K. Chopra, B. Zhai, D. Blaauw, and D. Sylvester, "A new statistical max operation for propagating skewness in statistical timing analysis," in *Proc. ICCAD*, San Jose, CA, USA, 2006, pp. 237–243.
[8] C.-Y. Chuang and W.-K. Mak, "Accurate closed-form parameterized block-based statistical timing analysis applying skew-normal distribution," in *Proc. ISQED*, San Jose, CA, USA, 2009, pp. 68–73.
[9] M. Merrett and M. Zwolinski, "Monte Carlo static timing analysis with statistical sampling," *Microelectron. Rel.*, vol. 54, no. 2, pp. 464–474, Feb. 2014.
[10] L. Zhang, C. C.-P. Chen, Y. Hu, and C. C. Chen, "Statistical static timing analysis with conditional linear MAX/MIN approximation and extended canonical timing model," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1183–1191, Jun. 2006.
[11] L. Xie and A. Davoodi, "Fast and accurate statistical static timing analysis with skewed process parameter variation," *IET Circuits, Devices Syst.*, vol. 2, no. 2, pp. 187–200, Apr. 2008.
[12] A. Singhee and R. A. Rutenbar, "Why quasi-Monte Carlo is better than Monte Carlo or Latin hypercube sampling for statistical circuit analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 11, pp. 1763–1776, Nov. 2010.
[13] V. Veetil, K. Chopra, D. Blaauw, and D. Sylvester, "Fast statistical static timing analysis using smart Monte Carlo techniques," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 6, pp. 852–865, Jun. 2011.
[14] 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. DAC*, Anaheim, CA, USA, Jun. 2010, pp. 793–798.
[15] C. Lemieux, "Using quasi–Monte Carlo in practice," in *Monte Carlo and Quasi-Monte Carlo Sampling*, 1st ed. Dordrecht, The Netherlands: Springer, 2009.
[16] M. Alioto, G. Scotti, and A. Trifiletti, "A novel framework to estimate the path delay variability on the back of an envelope via the fan-out-of-4 metric," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 8, pp. 2073–2085, Aug. 2017.
[17] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L_2 theory," *Zeitschrift Wahrscheinlichkeitstheorie Verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, 1981.
[18] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, Dec. 1979.
[19] K. R. Heloue and F. N. Najm, "Early statistical timing analysis with unknown within-die correlations," in *Proc. IEEE TAU Workshop*, Austin, TX, USA, Nov. 2007, pp. 1–6.
[20] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," in *Proc. ICCAD*, San Jose, CA, USA, 2003, p. 607.
[21] N. H. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. New Delhi, India: Pearson Education, 2015.
[22] J. H. Kim, W. Kim, and Y. H. Kim, "Efficient statistical timing analysis using deterministic cell delay models," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2709–2713, Nov. 2014.
[23] A. F. Gomez and V. Champac, "Critical path selection under NBTI/PBTI aging for adaptive frequency tuning," in *Proc. EWDTS*, Yerevan, Armenia, 2016, pp. 1–4.
[24] H. Chang and S. S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 9, pp. 1467–1482, Sep. 2005.
[25] W.-S. Wang and M. Orshansky, "Path-based statistical timing analysis handling arbitrary delay correlations: Theory and implementation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2976–2988, Dec. 2006.
[26] A. B. Kahng, "New game, new goal posts: A recent history of timing closure," in *Proc. DAC*, San Francisco, CA, USA, 2015, p. 4.
[27] C. Feng, R. Shyamsukha, S. Radhakrishnan, J. Zheng, and A. Gao, "Accelerating timing closure using incremental advanced OCV," in *Proc. CSTIC*, Shanghai, China, 2015, pp. 1–3.
[28] H. Fischer, "The central limit theorem from laplace to cauchy: Changes in stochastic objectives and in analytical methods," in *A History of the Central Limit Theorem*. New York, NY, USA: Springer, 2010, pp. 17–74.
[29] M. B. Wilk and R. Gnanadesikan, "Probability plotting methods for the analysis of data," *Biometrika*, vol. 55, no. 1, pp. 1–17, 1968.
[30] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, nos. 3–4, pp. 591–611, 1965.
[31] (2012). *Predictive Technology Model*. [Online]. Available: http://ptm.asu.edu
[32] S. H. C. Yen, D. H. C. Du, and S. Ghanta, "Efficient algorithms for extracting the K most critical paths in timing analysis," in *Proc. DAC*, Las Vegas, NV, USA, 1989, pp. 649–654.

HYUNJEONG KWON received the B.E. degree in electronic and electrical engineering from Pohang University of Science and Technology, Pohang, South Korea, in 2015, where she is currently pursuing the Ph.D. degree. Her current research interests include variation-aware circuit analysis methodologies.

JAE HOON KIM received the B.E. degree in electrical engineering from Hanyang University, South Korea, in 2009, and the Ph.D. degree in electrical engineering from the Pohang University of Science and Technology, South Korea. Since then, he joined Samsung Electronics Co., Ltd.

SEOKHYEONG KANG (S'11–M'13) received the B.S. and M.S. degrees in electrical engineering from the Pohang University of Science and Technology, Pohang, South Korea, in 1999 and 2001, respectively, and the Ph.D. degree from the VLSI CAD Laboratory, University of California at San Diego, San Diego, in 2013. He was with the System-on-Chip (SoC) Development Team, Samsung Electronics, Suwon, South Korea, from 2001 to 2008, where he was involved in development and commercialization of optical disk drive SoC.

He was a Professor with the Department of Electrical Engineering, Ulsan National Institute of Science and Technology, Ulsan, South Korea, from 2014 to 2018. He has been a Professor with the Department of Electrical Engineering, Pohang University of Science and Technology, since 2018. His current research interests include low-power design optimization and cost-driven methodology for chip implementation.

YOUNG HWAN KIM (S'86–M'89–SM'14) received the B.E. degree in electronics from Kyungpook National University, South Korea, in 1977, and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley, Berkeley, USA, in 1985 and 1988, respectively.

From 1977 to 1982, he was with the Agency for Defense Development, South Korea, where he was involved in various military research projects, including the development of auto-pilot guidance and control systems. From 1983 to 1988, he was a Post-Graduate Researcher with the Electronic Research Laboratory, University of California at Berkeley, where he was involved in developing VLSI CAD programs. He is currently a Professor with the Division of Electronic and Computer Engineering, Pohang University of Science and Technology, South Korea. His research interests include plasma and liquid crystal display systems, multimedia circuit design, MPSoC and GPGPU system design for display and computer vision applications, statistical analysis and design technology for deep-submicron semiconductor devices and power noise analysis. He has served as the General Chair and a Committee Member for various Korean domestic and international technical conferences, including the International SoC Design Conference, the IEEE ISCAS 2012, and the IEEE APCCAS 2016. He has served as an Editor for the *Journal of the Institute of Electronics Engineers of Korea*.