IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 15, 2020, accepted October 28, 2020, date of publication November 2, 2020, date of current version November 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035263

VASTA: A Wide Voltage Statistical Timing Analysis Tool Based on Variation-Aware Cell Delay Models

WENJIE FU, LEILEI JIN, MING LING, (Member, IEEE), YU ZHENG, AND LONGXING SHI, (Senior Member, IEEE)
National ASIC System Engineering Research Center, Southeast University, Nanjing 210096, China

Corresponding author: Ming Ling (trio@seu.edu.cn)

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61974024 and Grant 61874152.

ABSTRACT In the advanced technology nodes, process parameter variations are increasingly resulting in unpredictable device behavior. The issue is even aggravated by low power requirements which stretches the transistor operation into near-threshold regime. Despite device simulation gives precise results, it is time-consuming for static timing analysis and dynamic timing analysis. In this paper, we propose VASTA, a statistical timing analysis tool based on the variation-aware standard cell library. The tool efficiently supports statistical static timing analysis (SSTA) and statistical dynamic timing analysis (SDTA). The standard cell library models delay under operating environment effects by using quadratic regressions and multivariate adaptive regression splines. VASTA works on industry formats (*.v* and *.sdc*) and is designed to run in parallel during both SSTA and SDTA. The statistical cell library is built and verified using SMIC 40 nm and 28 nm PDK. The mean and standard deviation errors of cell delay models are 2.77% and 1.68% compared with SPICE simulation results under 10k Monte Carlo samples. The SSTA and SDTA are tested with ISCAS85, ISCAS89, and EPFL benchmark suites. The average mean and standard deviation errors of SSTA are 4.01% and 2.03%, which are similar to SDTA. Meanwhile, our SDTA is 17.7 times faster than traditional corner-based dynamic timing analysis which relies on generating *.vcd* cell activity files.

INDEX TERMS Delay model, dynamic analysis, process variation, statistical static timing analysis, voltage scaling.

I. INTRODUCTION

Timing analysis is one of the main procedures in the circuit design flow. It can be categorized into static timing analysis (STA) and dynamic timing analysis (DTA). STA checks the cell delays of the whole circuit to find out the worst arrival time of the output port for timing closure, regardless of input vectors of the circuit. DTA, on the other hand, characterizes the activities of paths by using input vectors and provides more optimistic timing results. It is mostly used in better-than-worst-case designs for design optimization to improve energy efficiency [1]. As a traditional industrial timing analysis tool, PrimeTime provides STA and DTA by using standard cell libraries under the given corner [2]. OpenTimer, an open-source static timing analysis tool, has a similar function [3]. It speeds up the analysis by parallel programming and reduces the timing slack by introducing common path pessimistic reduction techniques. However, both tools are highly dependent on the libraries that provide the cell delays indexed by the input slew and output load capacitance under a given supply voltage and temperature, which are not flexible in a voltage scaling scenario. Furthermore, the libraries are not accurate when the supply voltage scaled-down and process variations influencing the delay distribution more significantly. These inaccuracies may cause unpredictable device behavior in a low voltage scenario.

Considering process variation effects in the advanced technology nodes and low supply voltages, statistical STA (SSTA) and statistical DTA (SDTA) are proposed to replace traditional deterministic timing results with delay distributions [4], [5] by introducing random Gaussian variables. These analyses are performed by statistical cell delay models, which give the delay formulas with process variations for each cell by using linear regressions [6], quadratic

The associate editor coordinating the review of this manuscript and approving it for publication was Vivek Kumar Sehgal.

models [7], or multivariate adaptive regression splines (MARS) [8]. They have high precisions in modeling the circuit delay deviation under a specific operating condition (i.e., supply voltage and temperature). However, it is not general for different operating conditions. As the delay of each cell has a relationship with the operating conditions [9], it is not enough for the existing SSTA technology [4] to merely focus on specific conditions.

The lack of accurate and efficient timing analysis tools to support both STA and DTA considering process, voltage, and temperature (PVT) variations has been recently pointed out as a major weakness in timing analysis [10]. The tool must reflect such changes and update statistical timing information incrementally and accurately to ensure timing integrity as well as reasonable timing slack for voltage scaling [4]. Besides, due to the large computation time and memory requirements, existing dynamic timing analysis and optimization techniques are usually used in small designs [11], [12] [13]. So the tool needs an effective approach to accelerate the dynamic analysis for large designs.

In this paper, we propose VASTA, a high-efficiency statistical timing analysis tool that supports block-based statistical static and dynamic timing analysis. The analysis is performed by iteratively calculating the cell delays and output slew based on a variation-aware statistical standard cell library (slib). The slib quantifies the relationship between cell delays and PVT variations combined with quadratic regressions and MARS. In different supply voltage scenarios, our proposed slib can precisely estimate the cell delay and deviation without repeating SPICE and Monte Carlo analyses. The arrival time in SSTA and SDTA is estimated through statistical SUM and MAX operations. VASTA works on cell-level Verilog netlists and timing constraints files and is designed to run in parallel in both SSTA and SDTA. Besides, it can be easily combined with other advanced timing analysis techniques such as common path pessimistic reduction.

The main contributions of this paper can be summarized as follows:

1) Proposing a mapping model between delay model coefficients and operating conditions (e.g., supply voltages, temperatures, input slew, and output loads). The coefficients of mapping models compose a novel statistical library, slib. Given the cell type and operating conditions, the cell delay and output slew can be calculated based on the slib. Also, it greatly saves storage space without building different libraries for each corner.

2) Proposing statistical SUM and MAX operation solutions based on proposed statistical library (slib), which can give the statistical arrival time of each pin and output port of the circuit. Both SSTA and SDTA are designed to be run in parallel and are packaged into a tool. As the process of generating activity files (*.vcd*) is omitted in the tool, the computation time of the tool is significantly reduced.

The rest of this paper is structured as follows: Section II describes the modeling method of the cell delay distribution. Section III gives the details of the proposed statistical static and dynamic timing analyses. The experimental setup and the accuracy of VASTA are discussed in Section IV. Section V presents related works and Section VI concludes this paper.

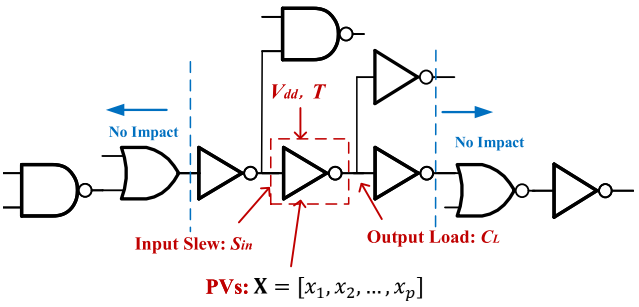


FIGURE 1. Key Components that Affect Cell Delay and Output Slew.

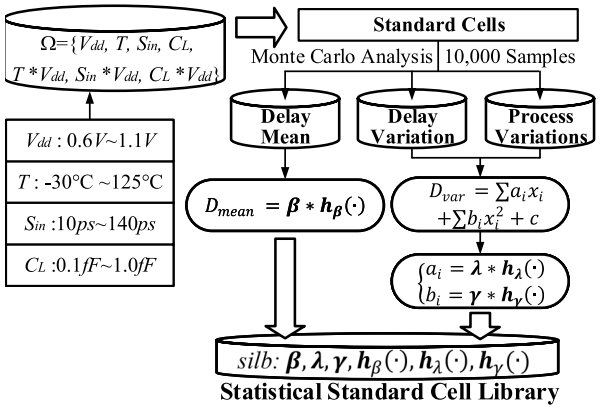


FIGURE 2. Variation-Aware Statistical Standard Cell Library Construction Flow.

II. STATISTICAL CELL LIBRARY ESTABLISHMENT

In library characterization, an accurate model for cell delay and output slew is developed given the following input data: a cell type, supply voltages (V_{dd}), temperatures (T), input slew (S_{in}), and output loads (C_L). The process variations (PVs) and operating conditions of each standard cell are considered during the cell delay distribution modeling, shown in Figure 1. The cell delay is determined by its input slew and output load, regardless of other cells that are not directly connected.

Figure 2 shows the modeling process of the statistical standard cell library (slib). The mean of the cell delay is considered as a function of operating conditions while delay variations are quantified by considering the PV influences [7], [14]. The operating conditions (V_{dd}, T, S_{in}, C_L) are also considered in [14] and in traditional industrial corner-based standard cell libraries [15]. The input sets for regressions are PVs, sampled from Monte Carlo simulations which are only required once for all standard cells under a specific technology node. The output set for regressions is the cell delay distribution captured by SPICE. Finally, the coefficients of our proposed cell delay models are stored in a complex lookup table, which can be accessed to get each cell's delay and deviation under any operating conditions. The

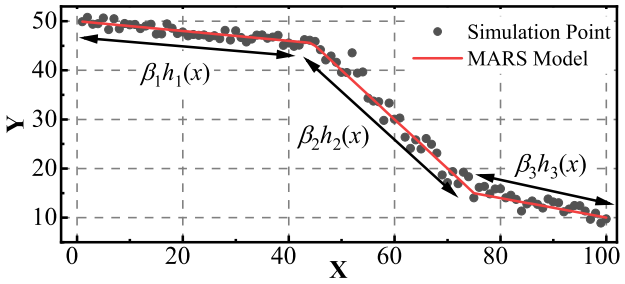


FIGURE 3. Variable Interaction in the MARS Model.

look-up table file is the statistical standard cell library, slib, under the given technology node.

In (1), we separate cell delay as mean value (D_{mean}) and mean-shift (D_{var}) which is called delay variation in this paper. If the delay formula is combined into a single model, the standard deviation is hard to be obtained for calculating statistical MAX in the block-based SSTA. Hence, the combined direct formula, $Delay = f(V_{dd}, T, S_{in}, C_L, PVs)$, can only be used for path-based SSTA proposed in the work [8].

$$D = D_{mean} + D_{var} \tag{1}$$

Firstly, we give the model for delay mean value considering the four operating conditions. To overcome the problem that operating conditions have an increasing influence on the mean of the delay, the influence of operating conditions is considered by using the MARS model. Instead of a predetermined form, MARS constructs the model structure by 'filtering out' the negligible variations without manual intervention with a form of hinge function pairs for variations, shown in Figure 3. Moreover, it captures essential nonlinearities and interactions. Consequently, it achieves high accuracy and is well suited for solving high-dimensional problems. The MARS model is conducted by hinge functions shown in (2), where β are constants, $h(\cdot)$ is the hinge function and n is the number of hinge functions. Since MARS performs analysis in parameter relative importance, the variables x_i with stronger impact on the result y_{MARS} will be modeled with more hinge functions to obtain higher accuracy, while the hinge functions of other variables will be less [16].

$$y_{MARS} = \beta_0 + \sum_{t=1}^{n} \beta_t * h_t(x_i) \tag{2}$$

To characterize the dependencies between each condition, the operating conditions are randomly sampled 10k times. The supply voltage V_{dd} ranges from 0.6V to 1.1V, the temperature T is -30℃$\sim$125℃, the input slew S_{in} is 10ps$\sim$140ps and the output load capacitance C_L is 0.1fF$\sim$1.0fF. The delay under these conditions is obtained through Monte Carlo simulations. Considering that the voltage has a strong coupling to the other three conditions, the hinge functions are built using the operation space $\Omega = \{V_{dd}, T, S_{in}, C_L, T*V_{dd}, S_{in}*V_{dd}, C_L * V_{dd}\}$, which can meet a higher precision in delay modeling comparing to the MARS model built by merely 4 elements V_{dd}, T, S_{in} and C_L. The mean of cell delay distribution is quantified by the MARS model, expressed through the operation space Ω, shown in (3). The number of hinge functions $h_t(\cdot)$ is decided automatically during the MARS modeling. The formula is rewritten as $\boldsymbol{\beta} * \boldsymbol{h_\beta}(\cdot)$ for simplification, which is stored in the slib as Figure 2 shows. Similarly, the output slew can be modeled in the same manner.

$$D_{mean} = \beta_0 + \sum_{t=1}^{n} \beta_t * h_t(\Omega) \sim \boldsymbol{\beta} * \boldsymbol{h_\beta}(\cdot) \tag{3}$$

In [7], a quadratic model was proposed to capture the nonlinearity between cell delay deviation D_{var} and PVs under a given operating condition, shown in (4). The $p*1$ Gaussian variation vector, $X = [x_1, x_2, \ldots, x_p]$, represents global variations such as threshold voltage variations, gate channel length variations and gate oxide thickness variations.[1]

$$D_{var} = \sum_{i=1}^{p} a_i x_i + \sum_{i=1}^{p} b_i x_i^2 + c \tag{4}$$

However, in different operating conditions, the delay deviation is different, which means the coefficients a_i, b_i and c are not constant values anymore. In this work, the coefficients a_i and b_i are represented by a function of operating conditions based on MARS models shown in (5). Vectors $\boldsymbol{\lambda} = \lambda_0, \lambda_1, \ldots, \lambda_n$ and $\boldsymbol{\gamma} = \gamma_0, \gamma_1, \ldots, \gamma_n$ are corresponding coefficients of every a_i and b_i formulas. Subsequently, the calculated a_i and b_i are taken back into (4) to get the deviation of the cell delay. As the mean of D_{var} is $\sum b_i + c$ and equals to zero, c is equal to the negative of $\sum b_i$.

$$\begin{cases} a_i = \lambda_0 + \sum_{t=1}^{n} \lambda_t * h(\Omega) \sim \boldsymbol{\lambda} * \boldsymbol{h_\lambda}(\cdot) \\ b_i = \gamma_0 + \sum_{t=1}^{n} \gamma * h(\Omega) \sim \boldsymbol{\gamma} * \boldsymbol{h_\gamma}(\cdot) \end{cases} \tag{5}$$

Ultimately, the cell delay can be obtained by calculating D_{mean} and D_{var} separately. All the coefficients $\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\gamma}, \boldsymbol{h_\beta}(\cdot), \boldsymbol{h_\lambda}(\cdot), \boldsymbol{h_\gamma}(\cdot)$ mentioned above are calculated and stored as the slib file in the look-up table form. So, the delay of each standard cell can be quantified for arbitrary circuit netlist under any operating condition.

III. STATISTICAL TIMING ANALYSIS IN VASTA

A. STATISTICAL STATIC TIMING ANALYSIS

The SUM and MAX are two fundamental operations to calculate the arrival times in the block-based timing analysis shown in Figure 4. Denoting that the arrival time of the input of a cell is D_1 and its delay is D_2, the arrival time of its output pin is the SUM operation of D_1 and D_2 and is equal to D_{sum}:

$$D_{sum} = D_1 + D_2 \tag{6}$$

If two timing arcs pointing to the same output pin of a cell, whose arrival times are D_3 and D_4, the arrival time of the

[1]In the SMIC 28 nm PDK, the PVs are defined as N_{vth0}, $N_{\mu 0}$, N_{voff}, N_{vsat}, P_{toxe}, P_{xl}, P_{xw}, P_{vth0}, $P_{\mu 0}$, P_{voff}, P_{vsat} and P_{eta0}.

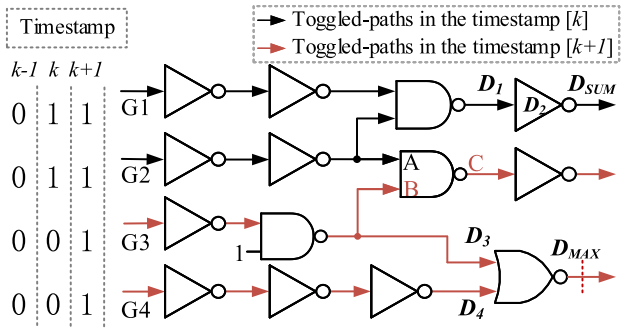


FIGURE 4. Toggled-Pins/Paths in the Dynamic Analysis.

output pin is the MAX operation of D_3 and D_4. Combined with the methodology proposed by Clark [17], the result of the MAX operation can be written as (7), where $D_{max,mean}$ is the maximum of two constant values $D_{3,mean}$ and $D_{4,mean}$. The coefficients $a_{m,i}$, $b_{m,i}$ and c_m can be computed as (8). $a_{D3,i}$, $b_{D3,i}$ and c_{D3} are coefficients in D_3 while $a_{D4,i}$, $b_{D4,i}$ and c_{D4} are coefficients in D_4. The symbols ϕ and φ are the cumulative distribution function (CDF) and the probability density function (PDF) of the standard Gaussian distribution calculated at $\mu_{D3-D4}/\sigma_{D3-D4}$. The values of μ_{D3-D4} and σ_{D3-D4} can be obtained in (9).

$$D_{max} = D_{max,mean} + \sum_{i=1}^{p} a_{m,i}x_i + \sum_{i=1}^{p} b_{m,i}x_i^2 + c_m \tag{7}$$

$$\begin{cases} a_{m,i} = \phi * a_{D3,i} + (1-\phi) * a_{D4,i} \\ b_{m,i} = \phi * b_{D3,i} + (1-\phi) * b_{D4,i} \\ c_m = \phi * c_{D3} + (1-\phi) * c_{D4} + \varphi * \sigma_{D3-D4} \\ \qquad + \phi * D_{3,mean} + (1-\phi) * D_{4,mean} - D_{max,mean} \end{cases} \tag{8}$$

$$\begin{cases} \mu_{D3-D4} = \sum_{i=1}^{p}(b_{D3,i} - b_{D4,i}) + (c_{D3} - c_{D4}) \\ \sigma_{D3-D4}^2 = \sum_{i=1}^{p}(a_{D3,i} - a_{D4,i})^2 + 2 * \sum_{i=1}^{p}(b_{D3,i} - b_{D4,i})^2 \end{cases} \tag{9}$$

B. STATISTICAL DYNAMIC TIMING ANALYSIS

Dynamic timing analysis identifies the transition state of each cell and gives the timing results in each input vector. The dynamic analysis in most cases shows a more optimistic result compared to that of static analysis. Before detailed description, we first give some definitions:

Definition 1: Timestamp: a timestamp gives the sequential input vectors of the test circuit. The number of timestamps is the number of input vectors.

Definition 2: Toggled-pin: a pin is toggled in the timestamp [k] when its value is changed compared to the value in the timestamp [k-1].

Definition 3: Toggled-path: a path is toggled in the timestamp [k] when the pins in the path are all toggled-pins.

Definition 4: Toggled-port: the ports are the input pins of the whole circuit. A port is toggled in timestamp [k] when the input stimulus of the port is changed compared to that in the timestamp [k-1].

Definition 5: Toggled-set: all the toggled paths in the given timestamp [k] compose a toggled-set.

As shown in Figure 4, the toggled-ports of the input vector in the timestamp [k] are G1 and G2, compared to the input vector in the timestamp [k-1]. Taking the pin A as an example, in the timestamp [k], pin A is toggled and requires time for its transition. However, the path followed by pin A is not sensitized because pin B is untoggled, which makes the state of pin C stay the same. The untoggled-pins do not have arrival time and those paths will be set as untoggled-paths. The toggled-set for the timestamp [k] as the input of SDTA consists of all the toggled paths. The delay for the target circuit is the arrival time of the toggled-set, which can be calculated using the slib and the SUM/MAX operations mentioned above. Similarly, the toggled-ports in the timestamp [k+1] are G3 and G4. In the timestamp [k], pin B is toggled and requires time for its transition which makes the state of pin C changed. Thus, in the timestamp [k+1], the arrival time of pin C is calculated regardless of pin A.

Algorithm 1 Calculating the Arrival Time of Each Pin in the Timestamp [k].

```
Input: Input Vector, Timing Graph
Output: Arrival Time AT [k]
1.  foreach timestamp [k] do
2.    reset_pins_as_untoggled ( );
3.    foreach pin p in timing graph do
4.      analyze_rise_fall_state ( );
5.      if state not equal to previous state do
6.        p->toggled ← true
7.      end if
8.    end for
9.    foreach pin p in timing graph do
10.     if p->toggled do
11.       SUM_MAX_operations ( );
12.       p->AT [k] ← MAX arrival time
13.     end if
14.   end for
15. end for
```

The pseudo-code for calculating the arrival time in each timestamp is presented in Algorithm 1. Firstly, the state of each pin is analyzed and compared to the state in the previous timestamp. If the state is changed, the pin is set to be toggled. After that, the arrival time of each pin will be calculated based on SUM/MAX operations.

Since the cell activity in the timestamp [k] is determined by the input vector in [k] and the states of each cell in [k-1], the SDTA can be parallelized after obtaining the cell states in the previous timestamp. A thread flipping method for multi-thread programming is proposed to accelerate the dynamic timing analysis speed shown in Figure 5. After getting the delay and slew of each cell, the input vectors will be fed

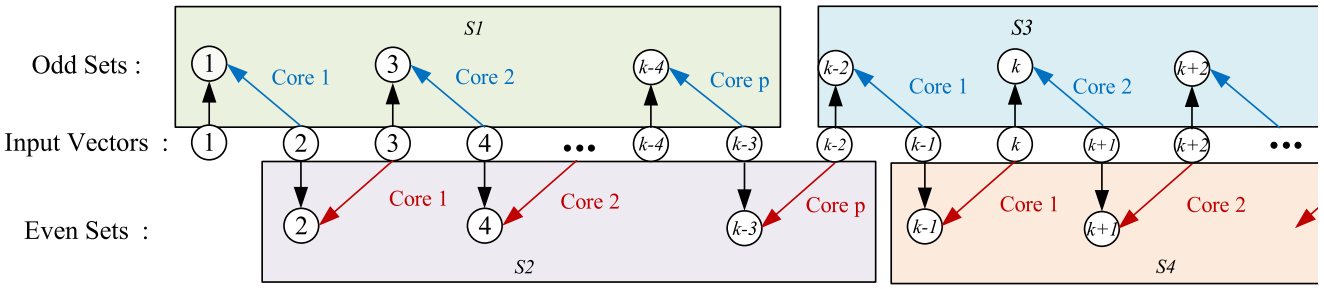


FIGURE 5. Timestamp-Based Efficient Thread Flipping Method in Dynamic Analysis.

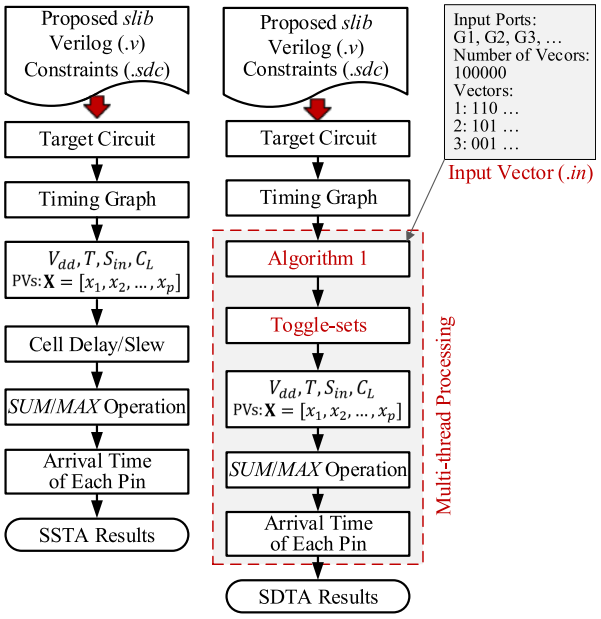


FIGURE 6. VASTA Workflow.

for SDTA and divided into odd sets and even sets. The odd set is firstly analyzed to mark whether the state of the pin is high or low. After that, the toggled pins of the even set will be identified compared to the states marked in the odd set and the arrival time of each pin will be calculated in parallel. Assuming that there are p cores calculated in parallel, p groups of comparisons can be performed at Stage 1 ($S1$). At the next Stage 2 ($S2$), another p groups of pin states can be analyzed. Afterward, the arrival time in odd sets will be calculated using information from even sets. Iteratively, the arrival time of all toggled-paths under each set of input vectors can be calculated for the target circuit. The multi-thread is realized using $C++$ ThreadPool library [18].

C. VASTA INTEGRATION

The workflow of the proposed timing analysis tool, VASTA, is shown in Figure 6. It reads a pre-established statistical library file (*.slib*) and several industrial format files both in SSTA and in SDTA. The industrial files include a Verilog netlist file (*.v*) and a timing constraint file (*.sdc*). Different from SSTA, a file including input vectors of the tested circuit is needed in the SDTA procedure.

While reading the netlist file, VASTA transmits the circuit netlist into a timing graph for timing analysis. The timing graph consists of timing arcs that show the connections between cell pins, including the pins from two connected cells and the pins from the input of the cell to its output. There are three member variables in each timing arc, which are the delay, slew, and arrival time. Based on the slib, the delay of each cell can be estimated by propagating the slew of each cell one by one. The delay, slew, and arrival time are propagated using the multi-thread method proposed in OpenTimer [3].

Assuming that the slew and delay of each cell will not be affected under input stimulus, the difference between realizing SSTA and SDTA is the arrival time of each pin. The calculated timing graph with cell delay and slew information can be reused between SSTA and SDTA. In SDTA, the parallel computing is started after getting the netlist timing graph. The pin whether toggled or not can be identified based on Algorithm 1. If not toggled, the arrival time of the pin will be set to -1 to reflect that the path with this pin is invalid. All the analyses from reading the input vectors of two sequential timestamps to getting the arrival time of each pin are in a single task. p tasks can be executed when the number of multi-thread is set to be p.

IV. EVALUATION

A. EXPERIMENTAL SETUP

The standard cells are modeled and verified using SMIC 40 nm and 28 nm PDK [20]. The supply voltage ranges from 0.6V to 1.1V while the temperature ranges from -30℃ to 125℃. The accuracy of timing analysis based on the proposed slib is verified using ISCAS85 benchmark suite [21], ISCAS89 benchmark suite [22] and EPFL benchmark suite [23]. ISCAS85 and ISCAS89 are well-known combinational and sequential benchmarks that are used in most studies [8], [13] [24]. EPFL benchmark is used in this work to verify the accuracy of larger circuits. The Verilog of benchmark netlists is generated through Design Compiler [25] using SMIC 28 nm PDK. VASTA is implemented in $C++$ on a 3.60GHz Intel i7-7700 Processor with 8 cores and 16GB memory. Both SSTA and SDTA in VASTA are executed with 8 threads concurrently.

B. THE CELL STATISTICAL MODELING ACCURACY

Under each technology node, the slib is built using 22 standard cells for simplification, which are INVV1/2/4/8,

TABLE 1. Average Modeling Error and Building Time of Standard Cells in 40 nm and 28 nm (0.6V ∼ 1.1V, 25℃).

Tech	Cell	[8]		[14]		[19]		slib		Model Build Time (s)		
		$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	[8]	[14]	silb
SMIC 40 nm	INV	3.12	4.39	8.83	3.05	2.04	0.51	2.56	1.02	155.96	11.56	756.02
	NAND2	2.80	3.85	8.30	4.10	4.66	3.60	2.28	1.82	313.88	36.62	1563.63
	NOR2	2.84	0.69	11.86	7.12	2.23	0.85	2.06	0.49	308.80	28.89	1517.00
SMIC 28 nm	INV	3.12	4.39	8.83	3.05	2.81	0.74	2.19	2.36	151.87	15.15	758.08
	NAND2	1.63	1.90	5.53	2.05	4.56	5.45	1.46	0.76	320.05	27.44	1610.15
	NOR2	1.94	0.22	8.96	4.22	3.33	1.61	1.53	1.21	294.21	28.70	1462.42
	XOR2	1.31	1.17	10.66	6.09	3.67	2.37	2.17	2.76	305.92	28.22	1508.62
	AOI22	1.58	1.14	9.38	4.81	5.97	4.78	2.13	1.79	578.99	46.87	2919.39
	OAI22	2.04	1.18	10.37	6.68	3.02	2.04	2.97	2.05	570.77	63.22	2872.26
	LAHQ	2.02	2.75	11.94	8.93	3.11	4.56	1.42	1.22	147.56	15.28	750.10
	DQ2	3.89	4.92	9.12	5.85	4.33	7.59	4.46	3.15	298.10	34.14	1486.42

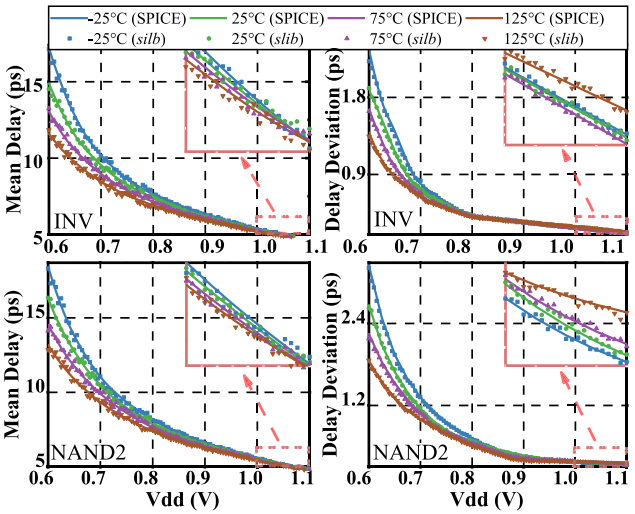


FIGURE 7. The Mean and Standard Deviation Comparisons Between slib and SPICE under MOS_MC Mode (28 nm).

NAND2V1/2/4/8, NOR2V1/2/4/8, XOR2V2/4, AOI22V2/4, OAI22V2/4, LAHQV2/4, and DQ2V2/4. These are the most basic standard cells and can be used for the logic synthesis of any circuit. In the following experiments, the supply voltage ranges from 0.6V to 1.1V, the temperature ranges from −30℃ to 125℃, the input slew ranges from 10ps to 140ps, and the output load capacitance ranges from 0.1fF to 1.0fF. These operating conditions are generated randomly in the cell model establishment. Figure 7 shows the mean and standard deviation of INV and NAND2 calculated using slib, compared to the delay distributions simulated by SPICE with 10k Monte Carlo samples under MOS_MC mode. The slib shows a high accuracy when the supply voltage decreases from 1.1V to 0.6V. The error is still within 3% even at 0.6V, indicating that our slib is suitable for not only normal voltage scenarios but also near-threshold voltage scenarios with high precisions. Figure 7 shows that the temperature has a significant effect on the delay deviation and is accurately captured by models in slib. The delay deviation of INV at 125℃ is much lower than that at other temperatures at 0.6V. While its delay deviation at 125℃ is higher at 1.1V. However, the delay distribution becomes more asymmetric with a long right tail when the supply voltage is scaled down to the sub-threshold region. As the delay model of this work is based on quadratic fitting, it models the mean and standard deviation value inaccurately in the sub-threshold region [26].

As shown in Table 1, the errors of our cell model are compared with several proposed delay models [8], [14] [19] under 40 nm and 28 nm. When the voltage is set from 0.6V to 1.1V with a step of 0.1V, the average mean and standard deviation errors stand by ε_μ and ε_σ are 2.77% and 1.68%, respectively. The errors at each supply voltage and temperature are calculated through (10). μ_{model} and σ_{model} are the mean and standard deviation of estimated delay distribution using proposed models while μ_{SPICE} and σ_{SPICE} are captured from SPICE Monte Carlo simulations as golden results. The results of [14] are worse than these shown in their paper since the temperature is a constant in their models, which is an important parameter demonstrated in Figure 7. As the slib is built under quadratic fittings and MARS, it is much slower in building the slib library comparing to other works. The library only needs to be created once for a given technology node. It is worthwhile to have a high precision library under different operating conditions, instead of the characterizing speed.

$$\varepsilon_\mu, \varepsilon_\sigma = \frac{abs(\mu, \sigma_{model} - \mu, \sigma_{SPICE})}{\mu, \sigma_{SPICE}} * 100\% \qquad (10)$$

C. PERFORMANCE OF VASTA

The ISCAS85, ISCAS89, and EPFL benchmark circuits are analyzed with VASTA based on the slib. The temperature and the output capacitance are set to be 25℃ and 0.1fF along with 10ps input slew. Figure 8 shows the timing results of c2670 benchmark under different supply voltages, including the corner-based delay, the 3σ points of delay distributions from SSTA and SDTA. The corner-based delay is simulated by SPICE with SS corner while SSTA and SDTA are performed by VASTA. The SDTA gives the distribution of 3σ delays at each voltage. In effect, the circuit workload decides the dynamic results of SDTA. As we compare the delay of the critical path under each input vector in error evaluation, the effect of workload to the error seems ignorable. To simplify the verification, we use 10k randomly generated input vectors. The X-axis at each voltage represents the ratio of the number of analyses with a delay t to the total number

TABLE 2. Average Error of SSTA and SDTA using ISCAS85, ISCAS89, and EPFL Benchmark Circuits at 0.6V, 0.8V, and 1.1V (25℃).

Benchmarks	Name	SSTA Error						SDTA Error					
		[8]		[24]		VASTA		[8]		[24]		VASTA	
		$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$	$\varepsilon_\mu(\%)$	$\varepsilon_\sigma(\%)$
ISCAS85	c432	8.11	3.87	2.86	2.48	4.39	2.17	7.70	3.68	4.63	3.90	4.20	2.08
	c499	7.45	3.08	4.12	3.58	3.94	1.98	7.20	2.93	5.05	4.24	3.87	1.91
	c880	7.99	4.01	3.49	4.32	3.52	2.00	7.57	3.80	4.08	3.43	3.39	1.92
	c1355	7.57	4.05	4.05	2.30	3.47	1.50	7.57	4.10	4.63	3.90	3.84	1.78
	c2670	7.78	3.32	4.34	3.56	1.62	0.78	7.60	3.71	3.60	3.03	4.01	2.08
ISCAS89	s208	8.15	4.56	3.45	2.85	4.08	4.38	7.59	4.31	3.59	3.02	4.77	3.92
	s386	9.63	4.19	4.90	2.07	3.82	2.90	8.27	4.39	4.59	3.85	3.75	2.60
	s820	5.74	3.09	4.61	3.68	4.73	1.24	5.08	2.79	3.56	2.99	2.68	1.91
	s1488	8.35	3.25	3.71	2.36	5.20	2.92	6.50	3.86	2.73	2.29	4.43	3.18
	s5378	6.81	4.10	4.98	4.11	3.28	2.30	5.16	4.66	3.08	2.59	5.59	2.84
EPFL	Adder	9.66	3.87	3.96	2.97	4.32	1.69						
	Bar	7.33	3.11	4.89	2.43	4.40	0.96						
	Max	6.52	2.68	2.68	4.23	2.86	1.83						
	Sine	7.86	3.51	4.02	2.56	4.19	1.35						

*As SPICE Monte Carlo simulations under 10k input vectors can hardly complete, errors of SDTA are not given in this table.

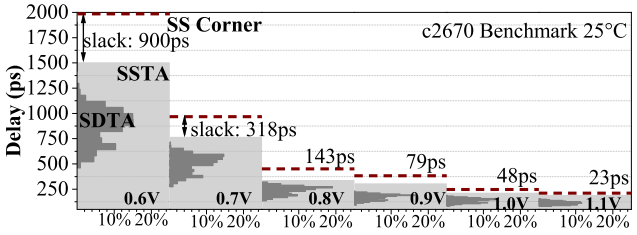


FIGURE 8. Delay Comparisons Between SS Corner, SSTA, and SDTA with 10k Random Inputs.

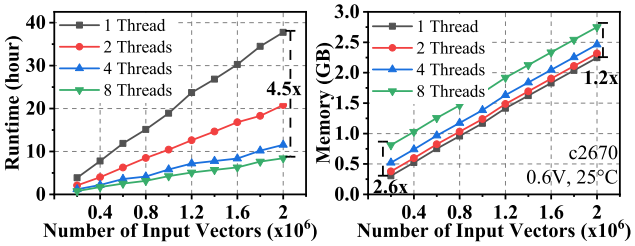


FIGURE 9. Runtime and Memory Usage for Multi-Thread.

of analysis times. The timing slack between worst delay and 3σ of SSTA is getting worse when the voltage is scaled down, which indicates that the timing analysis using traditional deterministic STA is too pessimistic and cannot provide precise results for low voltage circuit designs. Meanwhile, the dynamic 3σ delays under the given input vectors are much smaller than the SSTA results, let alone the corner-based results. The proposed VASTA can be used not only for statistical timing sign-off of low voltage designs but also for precise dynamic timing prediction for better-than-worst-case designs.

Table 2 gives average errors of proposed SSTA and SDTA at 0.6V, 0.8V, and 1.1V compared to a path-based SSTA with MARS-based cell models [8] using ISCAS85, ISCAS89, and EPFL benchmarks. The errors are also calculated using (10), where tested circuits for SPICE are whole netlists with specific input vectors. The cell delay models in slib and [8] for timing analyses are pre-built under 0.6V∼1.1V wide voltage range. The accuracy of a path-based SSTA [24] is also given in the table. Their Burr-based model cannot estimate the circuit delay precisely in such a wide supply voltage range. Compared with the results from SPICE Monte Carlo simulations, the mean error of VASTA SSTA is about 4.01% while the standard deviation error is about 2.03%. Due to the low precision of DFF delay models and the effect of the clock skewness, the error of sequential circuits (ISCA89) is slightly higher. The table also illustrates the mean and standard deviation errors of SDTA with 10k input vectors, which is similar to SSTA accuracies. Due to the huge computation and memory requirements for large circuits such as Adder, these circuits can hardly complete the dynamic timing simulations using SPICE and Monte Carlo analysis with thousands of input vectors.

The performance of parallel computing of SDTA is shown in Figure 9. The runtime scales down drastically with a single thread as the number of input vector increases. It can speed up about 4.5 times with 8 threads when the number of vectors increases to 2 million. The ratio is nearly a constant regardless of the number of vectors. The amount of peak memory usage is shown in the right of Figure 9. With the increase of thread count, more memory will be used for storing local variables in each thread. The absolute memory usage increment between 1 thread and 8 threads is a constant, about 500MB, which is related to the size of local variables, that is, the complexity of the tested circuit. When the vectors increase from 200k to 2M, the memory increment decreases from 2.6 times to 1.2 times. The ratio gets smaller because more memory is used for global variables for input vectors, other than local variables in each thread.

The proposed VASTA speedup is about 17.7 times, 3.6 times, and 3.1 times compared to the traditional DTA using PrimeTime, Graph-based DTA [11], and OpenTimer [3], shown in Figure 10. Traditionally, the activity file '*.vcd*' is needed and generated by VCS for DTA [27], which supplies a set of input vectors for DTA using either PrimeTime, Graph-based method, or OpenTimer. By analyzing the

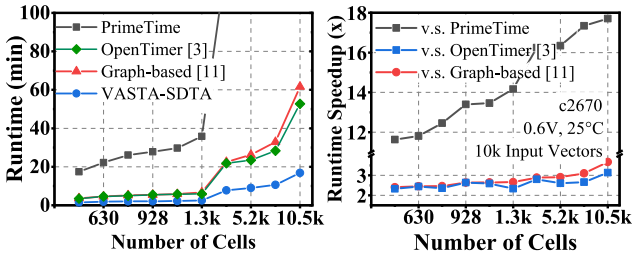


FIGURE 10. VASTA SDTA Speedup Compared to Traditional DTA using PrimeTime, Graph-based Method, and OpenTimer.

activity file, the untoggled-paths under each input vector are marked as an invalid path using command 'set_false_path' in PrimeTime and Graph-based method. In OpenTimer, it has a similar command for setting path invalid. Since VASTA can analyze and generate the timing graph considering the effect of input vectors, it can directly start SDTA to capture activity information of the cells and pins, rather than wasting time for generating *.vcd* activity files through RTL simulators. Figure 10 shows that VASTA merely uses several minutes to finish large circuit SDTA with 10k input vectors, while traditional timing analyses need hours. Meanwhile, VASTA greatly saves memory storage as the slib is suitable for different operating conditions, while traditional methods rely on thousands of corner files. Furthermore, VASTA SDTA effectively reduces time and memory overhead because it omits the process of generating activity files, which makes VASTA SDTA more suitable for large-scale circuit analysis.

V. RELATED WORK

It needs an exponential number of corner-based static timing analyses as the number of independent and significant sources increases. Some corner-based timing analysis tools like OpenTimer [3] and iTimerC2.0 [28] use an industrial liberty file, which fail to accurately characterize the effects of process variations in near-threshold voltage. Besides, they do not support SSTA, DTA, and SDTA.

Statistical STA methods are more realistic which treat the delay as a probability density function instead of constant values [6], [7]. The canonical quadratic model is widely used to parameterize statistical timing analysis which expresses the delay as a quadratic function of process variations [7]. It has high precision but is not general for different operating conditions [29]. Operating condition influence is considered by using the MARS model which exhibits higher precision and is well-suited for solving high-dimensional problems [8]. However, the single MARS form is not suitable for block-based analysis as the MAX operation in the analysis needs a definite form of the delay distribution.

SDTA with a proper set of vectors is much more accurate when the switching of gates is considered [12]. Existing methodologies in the area of DTA use pattern generation algorithms [13] or path-based techniques searching to get the worst-case delay. Graph-based techniques use toggle rates and statistical methods to find critical paths [11], [30]. Other attempts to incorporate voltage drop [31] by annotating pre-calculated voltages on STA or considers supply voltage as a global variable [32]. They lead to better results and miss the dynamically generated effects of the simulation and their interdependence, which are highly input pattern dependent.

VI. CONCLUSION

In this paper, we propose VASTA, a statistical timing analysis tool based on the variation-aware statistical standard cell library. The tool efficiently supports statistical static timing analysis and statistical dynamic timing analysis. The statistical standard cell library (*.slib*) models delay and operating environment effects by using quadratic regressions and multivariate adaptive regression splines. VASTA works on industry formats (*.v* and *.sdc*) and is designed to run in parallel in both SSTA and SDTA. The SSTA and SDTA are tested with ISCAS85, ISCAS89, and EPFL benchmark suites. The average mean and standard deviation errors of SSTA are 4.01% and 2.03%. The errors of SDTA are similar to SSTA. Meanwhile, our SDTA is 17.7 times faster than traditional corner-based dynamic timing analysis which relies on generating *.vcd* cell activity files.

REFERENCES

[1] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack redistribution for graceful degradation under voltage overscaling," in *Proc. 15th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2010, pp. 825–831.

[2] *PrimeTime Fundamentals User Guide, Version D-2010.06*, Synopsys Inc., Research Triangle, NC, USA, 2010. [Online]. Available: https://www.coursehero.com/file/59927684/Prime-Time-Fundamentals-User-Guide-201006pdf/

[3] T.-W. Huang and M. D. F. Wong, "OpenTimer: A high-performance timing analysis tool," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 895–902.

[4] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 4, pp. 589–607, Apr. 2008.

[5] J. Jung and T. Kim, "Variation-aware false path analysis based on statistical dynamic timing analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 11, pp. 1684–1697, Nov. 2012.

[6] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Be, J. Piaget, N. Venkateswaran, and J. G. Hemmett, "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.

[7] L. Zhang, W. Chen, Y. Hu, J. A. Gubner, and C. C.-P. Chen, "Correlation-preserved non-Gaussian statistical timing analysis with quadratic timing model," in *Proc. 42nd Design Autom. Conf.*, 2005, pp. 83–88.

[8] T. Liu, C.-C. Chen, and L. Milor, "Accurate standard cell characterization and statistical timing analysis using multivariate adaptive regression splines," in *Proc. 16th Int. Symp. Qual. Electron. Design*, Mar. 2015, pp. 272–279.

[9] M. Wagner and H.-J. Wunderlich, "Efficient variation-aware statistical dynamic timing analysis for delay test applications," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2013, pp. 276–281.

[10] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (NTV) design: Opportunities and challenges," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, 2012, pp. 1153–1158.

[11] H. Cherupalli and J. Sartori, "Graph-based dynamic analysis: Efficient characterization of dynamic timing and activity distributions," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 729–735.

[12] J.-J. Liou, A. Krstic, Y.-M. Jiang, and K.-T. Cheng, "Modeling, testing, and analysis for delay defects and noise effects in deep submicron devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 756–769, Jun. 2003.

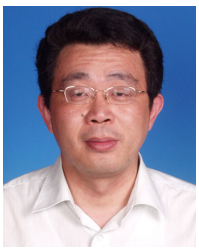
[13] B. Liu and L. Wang, "Dynamic statistical-timing-analysis-based VLSI path delay test pattern generation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1577–1590, Sep. 2015.
[14] L. Yu, S. Saxena, C. Hess, I. (. M. Elfadel, D. Antoniadis, and D. Boning, "Statistical library characterization using belief propagation across multiple technology nodes," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2015, pp. 1383–1388.
[15] C. Knoth, H. Jedda, and U. Schlichtmann, "Current source modeling for power and timing analysis at different supply voltages," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2012, pp. 923–928.
[16] W. Zhang and A. T. C. Goh, "Multivariate adaptive regression splines and neural network models for prediction of pile drivability," *Geosci. Frontiers*, vol. 7, no. 1, pp. 45–52, Jan. 2016.
[17] C. E. Clark, "The greatest of a finite set of random variables," *Oper. Res.*, vol. 9, no. 2, pp. 145–162, Apr. 1961.
[18] T.-W. Huang, C.-X. Lin, G. Guo, and M. Wong, "CPP-taskflow: Fast task-based parallel programming using modern C++," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2019, pp. 974–983.
[19] P. Cao, Z. Liu, J. Guo, and J. Wu, "An analytical gate delay model in Near/Subthreshold domain considering process variation," *IEEE Access*, vol. 7, pp. 171515–171524, 2019.
[20] SMIC. *SMIC Advanced Technology Overview*. Accessed: Oct. 2020. [Online]. Available: https://www.smics.com/en/site/technology_advanced_Te
[21] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test. Comput.*, vol. 16, no. 3, pp. 72–80, 1999.
[22] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, Jun. 1989, pp. 1929–1934.
[23] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," in *Proc. 24th Int. Workshop Log. Synth. (IWLS)*, 2015. [Online]. Available: https://infoscience.epfl.ch/record/207551/
[24] A. Moshrefi, H. Aghababa, and O. Shoaei, "Statistical estimation of delay in nano-scale CMOS circuits using burr distribution," *Microelectron. J.*, vol. 79, pp. 30–37, Sep. 2018.
[25] *Version c-2009.06*, DCU Guide, Synopsys, Mountain View, CA, USA, 2009.
[26] P. Cao, J. Wu, Z. Liu, J. Guo, J. Yang, and L. Shi, "A statistical current and delay model based on Log-Skew-Normal distribution for low voltage region," in *Proc. Great Lakes Symp. VLSI*, May 2019, pp. 323–326.
[27] *VU Guide*, Synopsys, Mountain View, CA, USA, 1998.
[28] P.-Y. Lee, I. H.-R. Jiang, C.-R. Li, W.-L. Chiu, and Y.-M. Yang, "ITimerC 2.0: Fast incremental timing and CPPR analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 890–894.
[29] W. Fu, Y. Zheng, L. Jin, and M. Ling, "A fast reduction method for path process variations without time-consuming training," in *Proc. IEEE 13th Int. Conf. ASIC (ASICON)*, Oct. 2019, pp. 1–4.
[30] H. Cherupalli and J. Sartori, "Scalable N-worst algorithms for dynamic timing and activity analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 585–592.
[31] R. Vishweshwara, R. Venkatraman, H. Udayakumar, and N. V. Arvind, "An approach to measure the performance impact of dynamic voltage fluctuations using static timing analysis," in *Proc. 22nd Int. Conf. VLSI Design*, Jan. 2009, pp. 519–524.
[32] B. Lasbouygues, R. Wilson, N. Azemard, and P. Maurine, "Temperature and voltage aware timing analysis: Application to voltage drops," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Apr. 2007, pp. 1–6.

WENJIE FU received the B.S. degree in microelectronics from the Wenzheng College, Soochow University, in 2016. He is currently pursuing the Ph.D. degree in electrical engineering. His main research interests include performance evaluation in computer architecture and statistical static timing analysis in low-voltage circuit designs.

LEILEI JIN received the B.S. degree in electronic information engineering from the China University of Petroleum, in 2018. She is currently pursuing the Ph.D. degree in electrical engineering. Her main research interest includes statistical timing analysis in low-voltage circuit designs.

MING LING (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Southeast University, in 1994, 2001, and 2011, respectively. He is currently working as an Associate Professor with the National ASIC System Engineering Technology Research Center, Southeast University. His main research interests include memory subsystem of SoC, embedded software, and SoC architecture.

YU ZHENG received the B.S. degree in electronic information from Tianjin University, in 2018. He is currently pursuing the M.D. degree in electrical engineering. His main research interest includes statistical timing analysis in wide-voltage circuit designs.



LONGXING SHI (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Southeast University, Nanjing, China, in 1984, 1987, and 1992, respectively, all in electronic engineering. He is currently a Professor and the Dean of the National ASIC system Engineering Research Center, Southeast University. His research interests include system-on-a-chip design, VLSI design, and power IC design.