SD-SSTA: Statistical Static Time Analysis Algorithm Considering Skewed Distribution

Fuxing Deng[1*],Yihang Feng[1*], Dan Niu[2], Xiao Wu[3] and Zhou Jin[1]
1.Super Scientific Software Laboratory, China University of Petroleum-Beijing, Beijing, China
2.School of Automation, Southeast University, Nanjing, China
3.Huada Empyrean Software Co. Ltd, Beijing, China
Email: fuxing.deng@student.cup.edu.cn, yihang.feng@student.cup.edu.cn,
101011786@seu.edu.cn, wuxiao@mail.empyrean.com.cn, jinzhou@cup.edu.cn

***Abstract*—Static Timing Analysis (STA) is one of the most widely used and successful analysis engines in digital circuit design in recent years. However, the Deterministic Static Timing Analysis (DSTA) does not take into account the effect of process parameter variability on circuit performance, which arouses people's attention to the ability of STA to effectively simulate statistical changes. Therefore, Statistical Static Timing Analysis (SSTA) has been proposed and extensively studied. Traditional SSTA algorithms, such as probabilistic propagation based on Gaussian distribution and Monte Carlo simulation, cannot achieve a high accuracy and good performance. In this paper, a SSTA algorithm considering skew distribution, SD-SSTA, is proposed, which successfully realizes accurate calculation of arrival time and timing margin, and has excellent performance.**

The paper makes three contributions. (1) We convert the non-Gaussian distribution into a Gaussian Mixture Model (GMM), which fits the real result better than the traditional SSTA algorithms. (2) We consider the influence of skew and introduce *Skew Adjustment Factor* (*SAF*) into the calculation of timing margin to ensure that the results are more realistic. (3) We use the name mapping method to reduce the memory consumption of the algorithm, which further improves the algorithm memory performance. Compared with SSTA algorithm based on Gaussian distribution, SD-SSTA algorithm has excellent performance in both accuracy and performance.

***Index Terms*—Statistical Static Timing Analysis, non-Gaussian, Gaussian Mixture Model, Timing Margin**

I. Introduction

With the rapid development of integrated circuit (IC) technology, timing becomes more and more critical in modern IC design. Static Timing Analysis (STA) is a critical step in ensuring the successful tapeout of IC. One reason STA tools are widely used is that their algorithmic complexity is linearly related to the complexity of IC design. This feature ensures that STA tools can efficiently analyze ICs with reasonable time and memory requirements [1].

Traditional STA tools assume that the manufacturing process parameters (such as ion implantation concentration, oxide thickness, etc.) are fixed values [2], so the logic gate and interconnect delay is a constant. However, this is obviously inconsistent with the actual situation, due to the limitations of various factors such as equipment, even if it is the same type of multiple logic gates on the same chip, its delay parameters can not be exactly the same. To introduce this uncertainty, designers often introduce a pessimistic quantity into traditional STA tools, and under advanced manufacturing processes, choosing a pessimistic quantity has become increasingly difficult [3], [4]. In order to more realistically simulate this uncertainty, a more accurate Statistical Static Timing Analysis (SSTA) is proposed.

In current SSTA research [5]–[7], various methods have emerged to estimate the timing performance of circuits more accurately. The probability propagation method can effectively calculate the output probability distribution by propagating the probability distribution of the mutation source to the sequential path of the circuit [8], [9]. In order to facilitate analysis and shorten running time, Gaussian distribution is always the main choice of statistical distribution. However, in the actual manufacturing process, due to the complex correlation and nonlinear effects between the process parameters, the arrival time of the nodes may not be Gaussian distribution. In these cases, then, Monte Carlo simulation seems to be a promising method for dealing with arbitrary distributions [10], which improves the treatment of variability through random sampling of process parameters. However, Monte Carlo simulation usually requires a large number of random sampling and simulation, resulting in high computational complexity. This can become less practical for timing analysis of large-scale IC, especially when multiple process parameters need to be considered [11]. Therefore, to maintain the high accuracy and good performance simultaneously is non-trivial.

In the paper, we propose **SD-SSTA** algorithm and name mapping method to resolve these problems. (1) We propose a SSTA algorithm considering non-Gaussian distribution. The algorithm uses Monte Carlo method for max operations on non-Gaussian distributions, and then converts the non-Gaussian distribution into the GMM.Azuma et al [12] give closed formulas for the parameters of the GMM so as to realize the forward propagation of the non-Gaussian distribution and calculate the arrival time more accurately. Further, we introduce ***SAF*** into the formula for calculating the timing margin. The value of ***SAF*** is affected by two factors, one is the skewness and the other is the variance of the delay. For different circuits, we dynamically adjust the scale parameter in ***SAF*** according to the variance size of the relevant circuit

[*]Both authors contributed equally to this research.

design file. The purpose is to ensure that the calculation of timing margin is more realistic. (2) We propose a method of name mapping to reduce the memory requirement of the algorithm. In the circuit design file, node names are recorded as string types. For large-scale circuits, these names are often long, so reading them into a data structure can take up a lot of memory space, and the time to compare two string variables is significantly greater than the time to compare two integer variable. By mapping string type to integer type, the memory can be reduced effectively and the time can be shortened.

In the experimental part, we compare the **SD-SSTA** algorithm with the traditional SSTA algorithm [13], and evaluate the accuracy of their timing margin at the endpoints. The experimental results show that the **SD-SSTA** algorithm not only maintains high accuracy, but also exhibits excellent time and space efficiency.

II. Preliminaries

A. Mathematical Preliminaries

In this paper we use the following explicit notation for the Gaussian probability density functions (PDFs):

$$\phi(x|\mu,\sigma) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}\sigma}\varphi\left(\frac{x-\mu}{\sigma}\right),\ \varphi(x) \stackrel{\text{def}}{=} e^{-\frac{1}{2}x^2} \tag{1}$$

where the function φ(x) is referred to as the Gaussian kernel. The Cumulative Distribution Function (CDF) of the standard Gaussian distribution will be denoted as follows:

$$\Phi(x|\mu,\sigma) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}\varphi(t)dt \tag{2}$$

B. Arrival Time and Timing Margin

Sequential circuits refer to circuits in digital circuits that have specific timing requirements, and operate in a certain order under the control of a specific clock signal. In the design and analysis of sequential circuits, there are two important concepts: *Arrival Time* and *Timing Margin*.

1) Arrival Time: Arrival time refers to the time required for a specific point in a circuit to receive a signal from other logic units. This time encompasses the cumulative delay along all paths through which the signal propagates to reach the given point.Accurate calculation of the arrival time is essential to ensure the correct function of the circuit and to meet timing requirements.

2) Timing Margin: The timing margin is typically defined as the time margin before or after a signal reaches the target component, representing the time difference around its intended processing time. In general timing analysis, it encompasses both setup time and hold time, ensuring that the signal arrives and stabilizes within the specified time window to guarantee proper circuit operation.

C. Statistical STA

In the SSTA, the key parameters in the circuit, such as logic gate delay and signal arrival time, are modeled as a Gaussian distribution. This is a simplified assumption. For example, [14], [15] assume that all sources of change are Gaussian and independent of each other.

In order to calculate the arrival time and timing margin in the block-based SSTA framework, two atomic operations are required, namely *Add* and *Max*. A similar discussion applies to subtraction and minimization.

1) Add Operation: The arrival time at the cell output is calculated by adding the input arrival time to the cell delay. For SSTA, the arrival time and cell delay are random variables (RVs) that obey the Gaussian distribution, so the *Add* operation is essentially the addition of the Gaussian distribution.

$$P = \sum P_i\left(u_i,\sigma_i\right) \cong N\left(\mu,\sigma\right) \tag{3}$$

$$\mu = \sum_{i=1}^{n}\mu_i \quad and \quad \sigma = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\rho_{ij}\sigma_i\sigma_j} \tag{4}$$

N(μ,σ) is Gaussian with mean μ and sigma σ and P_i is the delay distribution for each individual cell on the path and ρ_{ij} is the correlation coefficient between delays of cells i and j.

2) Max Operation: the maximum operation is required for multiple input units, where the worst-case output arrival time is the maximum of all output arrival times.

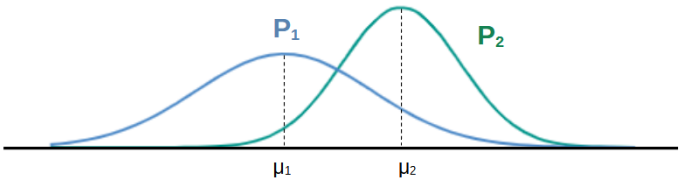

Fig. 1: For two independent normal distributions with different means and variances, the *Max* operation depends mainly on their means.

Suppose that to obtain the maximum values of two independent Gaussian distributions P_1 and P_2 as shown in Figure 1, the mean are μ_1 and μ_2, respectively, and the standard deviation are σ_1 and σ_2 respectively. SSTA algorithm use μ + $N\times\sigma$ (N is constant.) as the comparison condition. The result of the *Max* operation is given by

$$\mu = max(\mu_1,\mu_2) \tag{5}$$

$$\sigma = \frac{max(\mu_1+N\times\sigma_1,\mu_2+N\times\sigma_2)-\mu}{N} \tag{6}$$

D. Gaussian Mixture Model (GMM)

A GMM can be regarded as a model composed of K individual Gaussian models, where these K models serve as the hidden variables of the mixture model. In general, any probability distribution can be used in a mixture model, but GMMs are chosen because Gaussian distributions possess excellent mathematical properties and efficient computational performance. They can smoothly fit PDFs of arbitrary shapes. GMM is widely used in signal processing [16] and machine learning [17], [18]. A simple example is shown in Figure 2, where the gray area represents the GMM formed by the two

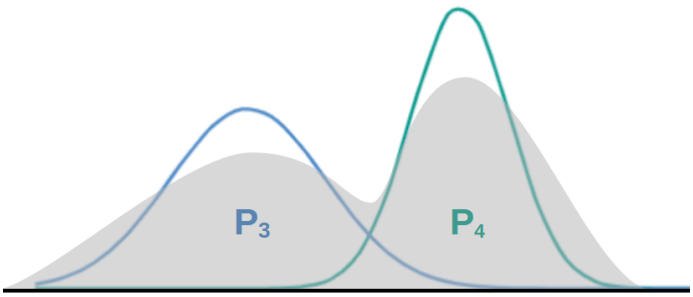

Fig. 2: A GMM consists of two Gaussian distributions.

Gaussian distributions P_3 and P_4. And The mathematical form of GMM is as equation 7. Such basis function $\phi(x)$ is called Radial Basis Function (RBF).

$$f_{GMM}(x) = \sum_{i=1}^{N} w_i \cdot \phi(x|\mu_i, \sigma_i) \tag{7}$$

E. Skewness

In practical scenarios, due to variations in manufacturing processes, environmental factors, etc., arrival times may exhibit a certain degree of asymmetry. Therefore, using skewed distributions can more accurately reflect real-world situations and improve modeling accuracy. Skewness is a measure of the degree to which a probability distribution deviates from symmetry in probability statistics. The skewness of a probability distribution is quantified by statistical measures, with the most common measure being the *skewness coefficient*. In computer science, kernel density estimation, quantile regression, and mixture models are commonly used for modeling skewed distributions [19]–[21].

III. Algorithm Frameworks

A. Overall Framework

Our proposed **SD-SSTA** algorithm is divided into the following three stages. (1) Read the *circuit design file* and establish the data structure of the timing diagram. (2) Forward propagation is carried out, and the arrival time of all nodes is gradually calculated from the root nodes of the timing diagram. (3) Calculate timing margin at the endpoints of the specified timing path. Figure 3 shows the flow of **SD-SSTA** algorithm.

B. SD-SSTA

1) Build Timing Diagram: We collect statistical data of circuit elements and organize circuit design files, which include various important parts describing circuit structure and timing information. A simple circuit example is shown in Figure 4, where the **SD-SSTA** algorithm constructs the timing diagram shown in Figure 5 by reading the circuit design file.

2) Operations for SD-SSTA: The initial inputs are Gaussian distributions, add the input and delay of the logic gate. The distribution of the logic gate output is shown as follows

$$f_{gate} = Max(X_1, X_2) + X_0 \tag{8}$$

where X_1 and X_2 are the RVs that describe the arrival time of input signals, and X_0 is the RV that gives the gate operation time. The logic gate delay often has Gaussian distribution [22].

In this paper, the non-Gaussian distribution is described by adding *skewness coefficient* to the Gaussian distribution. So the calculation of arrival time by **SD-SSTA** algorithm involves two parts.

a) μ and σ: A non-Gaussian distribution is usually formed after **Max** operation. So we propose to model the PDF of the max with GMM. First, we use Monte Carlo simulation to model non-Gaussian distributions, and then fit it using GMM. In other words, it can be decomposed into RBFs [23]. Furthermore, we extract the μ and σ of each fitted RBF from the model's attributes. As thus we handle non-Gaussian distributions as linear combinations of functions of the equation 1, as schematically shown in Figure 6. Finally, μ and σ of arrival time are extracted by the following formula:

$$u_{gate} = \sum_{i=1}^{N} w_i \cdot \mu_i \quad and \quad \sigma_{gate} = \sqrt{\sum_{i=1}^{N} w_i \cdot \sigma_i^2} \tag{9}$$

where N, w_i, μ_i and σ_i corresponds to formula (6) one by one.

b) skewness coefficient: The PDF for the maximum of two correlated Gaussian RVs ($X_1 \, and \, X_2$) has the form:

$$\begin{aligned} f_{max}(x,\rho) = f_1(x)\Phi\left[\frac{1}{\sqrt{1-\rho^2}}\left(\frac{x-u_2}{\sigma_2} - \rho\frac{x-u_1}{\sigma_1}\right)\right] \\ + f_2(x)\Phi\left[\frac{1}{\sqrt{1-\rho^2}}\left(\frac{x-u_1}{\sigma_1} - \rho\frac{x-u_2}{\sigma_2}\right)\right] \end{aligned} \tag{10}$$

where f_i(x) is a Gaussian PDF the i^{th} distribution (i = 1,2) respectively.

The third moment is a measure of skewness. Therefore, by calculating the third moment, we can describe the skewness of a non-Gaussian distribution after the **MAX** operation. The calculation formula is as follows

$$skew = E\left[(X-\mu)^3\right] = \int_{-\infty}^{\infty} (x-\mu)^3 f_{max}(x)dx \tag{11}$$

Algorithm 1 describes the operations for **SD-SSTA** using pseudo-code.

From the pseudo-code, we can see that the **SD-SSTA** algorithm uses the idea of *topological sorting* to calculate the arrival time. Based on the timing diagram, the arrival time is calculated from the root nodes from shallow to deep, which is called *forward propagation*. Each iteration computes the current root nodes and then updates the root nodes as the root nodes for the next iteration.

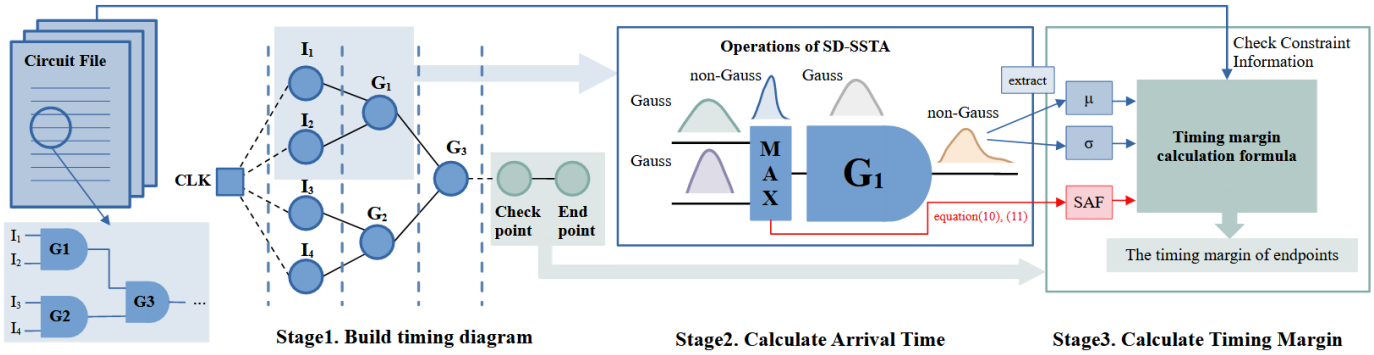

Fig. 3: Implementation process of **SD-SSTA** algorithm.

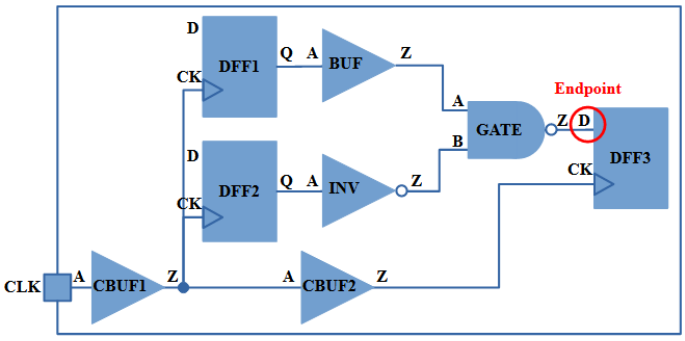

Fig. 4: A simple example of a sequential circuit.

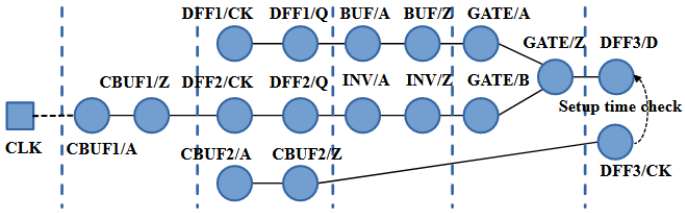

Fig. 5: Visualization of data structure for timing diagram.

3) Calculation of Time Margin: **SD-SSTA** algorithm calculates the timing margin at the endpoints of the specified timing path. For the trigger triggered by the rising edge as shown in Figure 4, timing margin of the falling signal at DFF/D in Figure 4 applies the following formulas to calculate.

$$reqTime = CP + u_{\text{min,rise}} - N \times \sigma_{\text{min,rise}} - SetupValue_{\text{fall}} \quad (12)$$

$$arrTime = u_{\text{max,fall}} + N \times \sigma_{\text{max,fall}} \quad (13)$$

$$\boldsymbol{SAF} = skew \times k \quad (14)$$

$$Timing\ margin = reqTime – arrTime + \boldsymbol{SAF} \quad (15)$$

N is the specified confidence interval. CP is the clock period. $\mu_{\text{max/min,rise/fall}}$ and $\sigma_{\text{max/min,rise/fall}}$ mean that on the node, the data jump direction is rise/fall, and the mean and sigma values in the max/min environment. $SetupValue_{\text{fall}}$ is the value from node n_1 to n_2, and the jump direction of n_2 is the establishment time requirement of rise/fall. k is the scale factor, which value is dynamically adjusted according to the size of the standard deviation of the distribution. ***SAF*** is a skew adjustment factor whose value varies according to the positive or negative and magnitude of PDF skew. For positive skewness, ***SAF***>0 and for negative skewness, ***SAF***<0.

Algorithm 1 Operations for SD-SSTA

Input: graph G and delay distributions for nodes and input signals
Output: the arrival time of all nodes

```
Topologically sort graph G; # Every time we get G' (all the nodes with zero degree.)
for node in G' do
    node ← initialise;
    if number_predecessor nodes>1 then
        for predecessor nodes of the node do
            f_mc ← utilise the Monte Carlo;
            f_GMM ← decompose f_mc into RBFs;
        end for
    end if
    if number_predecessor nodes=1 then
        f_GMM ← f_gate of predecessor node;
    end if
    f_gate ← f_GMM + f_0;
end for
```

C. Performance Optimization

In STA, names of a logic gate are usually stored in the circuit file in the form of strings. For large-scale circuits, the search and comparison of strings will consume a lot of memory and time. We propose a *name mapping method* to map all names of a logic gate in the circuit file into integer numbers, and only use integer numbers for calculation during the operation of the program. This way, especially in large-scale circuits, saves a lot of memory, and also reduces the running time to a certain extent. Figure 7 shows the mapping process for the Hash function.

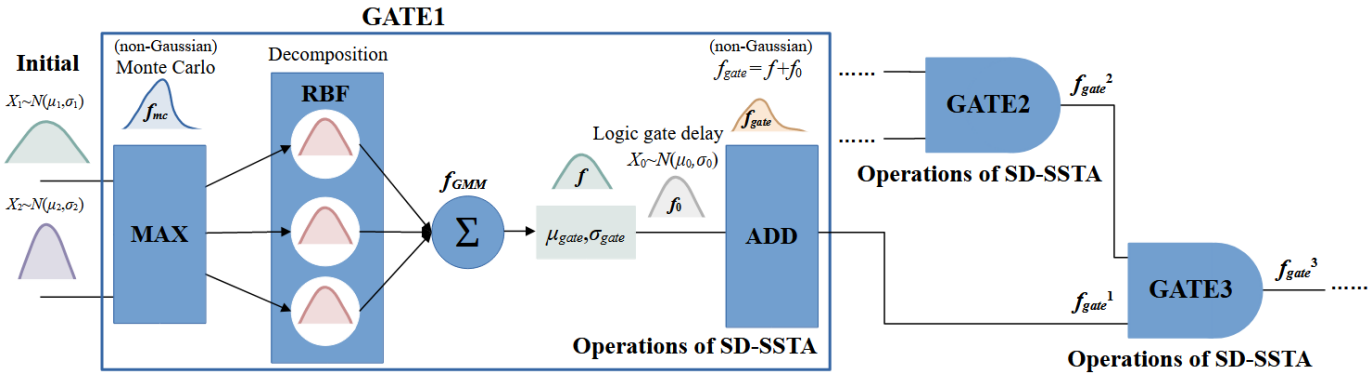

Fig. 6: Implementation of non-Gaussian distributed Max and Add operations.

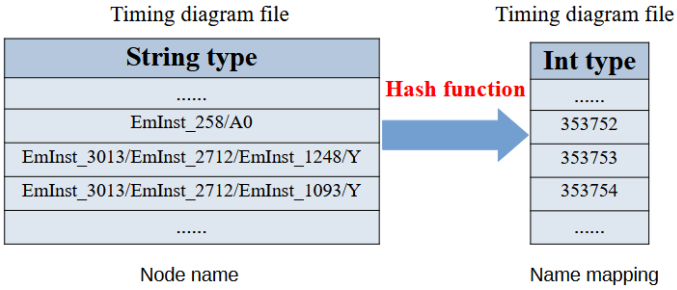

Fig. 7: Name mapping method.

IV. Experiment Results

A. Experimental Setup

In order to improve the computational efficiency, the paper assumes that each unit delay is independent from each other, without considering the correlation between them. So ρ_{ij}=0 for i≠j, equation 4 reduceds to 16 and equation 10 reduceds to 17.

$$\mu = \sum_{i=1}^{n} \mu_i \quad and \quad \sigma = \sqrt{\sum_{i=1}^{n} \sigma_i^2} \tag{16}$$

$$f_{max}(x) = f_1(x)\Phi\left(\frac{x-u_2}{\sigma_2}\right) + f_2(x)\Phi\left(\frac{x-u_1}{\sigma_1}\right) \tag{17}$$

To have a higher level of confidence, we set the constant N of equation 12 and equation 13 to 3.

The paper uses *GNU time* to supervise the calculation of arrival time and timing margin by two algorithms.

1) Environment Configuration: The specific experimental environment configuration of this paper is shown in table I. The tool implementation uses python3, python version is 3.8.1.

TABLE I: Environment configuration.

CPU	AMD EPYC 7702 64-Core Processor
System	Ubuntu
Architecture	x86_64
Thread(s) per core	2

2) Test cases: The paper uses three test cases for the experiment,the data came from the 2023 of the integrated circuit eda elite challenge, each case containing three circuit design files. (1) *Timing diagram file*,which stores sense,max rise delay mean, max rise delay sigma, max fall delay mean, max fall delay sigma, min rise mean , min rise delay sigma ,min fall delay mean, min fall delay sigma data; (2) *Setup time check file*, which records all check constraint information includes sense rise constraint ,fall constraint data; (3) *End-points list*, which records the endpoints of all timing paths. The information of different cases is shown in Table II.

TABLE II: Information about the test cases. The first two lines reflect the data size of the cases (The values in the first row can reflect the number of nodes and edges in the timing diagram. The values in the second row represent the number of endpoints.). The values in the third row is the mean of the standard deviations of the various cases.

	Circuit 1	Circuit 2	Circuit 3
Timing diagram file	7894	8794	126998
Endpoints list	1313	1313	4502
Mean of σ	0.004047	0.036749	0.000744

3) Evaluation index: In the paper, we use the results of equation 20, $score_{total}$, as an evaluation index. It can measure the accuracy of **SD-SSTA** algorithm to calculate the timing margin of all endpoints.

$$err_{single} = abs\left(\frac{our\ results - standard\ result}{standard\ result}\right) \tag{18}$$

$$score_{single} = \begin{cases} 1 - err_{single} & , err_{single} \leq 20\% \\ 0 & , err_{single} \leq 20\% \end{cases} \tag{19}$$

$$score_{total} = 100 \times \frac{\sum_{all\ results} score_{single}}{number\ of\ results} \tag{20}$$

For a single endpoint, the error of the result is calculated as in equation 18, and its score is calculated as in equation 19. When the error of a single endpoint exceeds 20%, the result is invalid, and the endpoint score is zero. For all endpoints, as in equation 20, full score are 100.

B. Accuracy result

Based on equation 20, we calculate the corresponding timing margin accuracy score. Table III compares the timing margin calculated by SSTA [13] and **SD-SSTA** algorithms for three different test cases.

TABLE III: Final score for each case.

Method	Accuracy score of different cases		
	Circuit 1	Circuit 2	Circuit 3
SSTA	97.450931	80.584524	99.425688
SD-SSTA	99.477106	98.251913	99.861775

As can be seen from the table, the variance of circuit 3 is too small, that is, the variation range of process parameters is small, so the traditional algorithm can achieve an accurate calculation. The variance of circuit 2 is too large, a larger variance means that the Gauss distribution is more flat, which means that the process parameters vary in a wide range, the traditional SSTA in this case, often lead to greater error and the superiority of **SD-SSTA** algorithm becomes apparent. All in all, **SD-SSTA** algorithm has higher accuracy than the traditional algorithm implemented with Gaussian distribution.

C. Time and Memory performance analysis

In this paper, the "Elapsed (wall clock) time" from the $GNU\ time$ output results is chosen as the runtime, and the "Maximum resident set size" is used to represent the peak memory usage. We compare the algorithm using the name mapping method with the original algorithm, and the efficiency of optimization is shown in Figure 8.

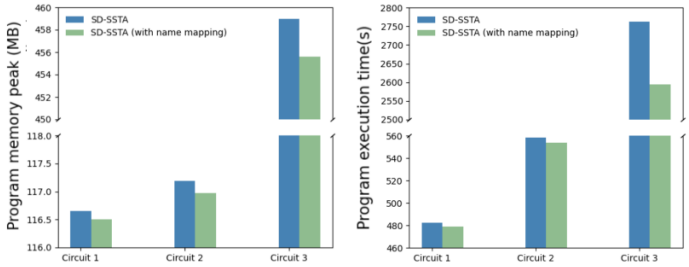

Fig. 8: Time and memory comparison before and after the name mapping method is used.

As can be seen from the Figure 8, the name mapping method reduces the time and memory of the algorithm, and its effect is especially obvious for large-scale circuits.

Conclusion

In this paper, a non-Gaussian distribution based SSTA method, **SD-SSTA**, is proposed. Different from the traditional SSTA, we use GMM to model non-Gaussian distribution, and introduce *skewness coefficient* on the basis of Gaussian distribution. A non-Gaussian distribution is represented by μ, σ and *skewness coefficient*, that is, the non-Gaussian distribution is parameterized. Using these parameters, the forward propagation of arrival time is calculated accurately. In the calculation of timing margin, we consider the influence of skew and introduce ***SAF*** into the formula. Compared with the traditional SSTA results, the **SD-SSTA** algorithm has significantly improved the accuracy of the calculation of timing margin. Further, the memory and time are reduced effectively and the performance of the algorithm is improved by name mapping.

Acknowledgment

We are deeply grateful to the reviewers for their valuable comments. Zhou Jin is the corresponding author of this paper. This work was supported by the Natural Science Foundation of China (Grant Nos. 62204265, 62234010, 62374031) and the National Natural Regional Key Fund (Grant No. U23A20301).

References

[1] S. H. Gerez. *Algorithms for VLSI Design Automation*. Wiley, 1998.

[2] D. Blaauw, K. Chopra, A. Srivastava, and L. K. Scheffer. Statistical timing analysis: From basic principles to state of the art. *IEEE TCAD*, 2008.

[3] S. Nassif. Within-chip variability analysis. In *IDEM '98*, 1998.

[4] S. R. Nassif. Modeling and analysis of manufacturing variations. In *CICC '01*, 2001.

[5] V. Khandelwal and A. Srivastava. A general framework for accurate statistical timing analysis considering correlations. In *DAC '05*, 2005.

[6] J. Singh and S. Sapatnekar. Statistical timing analysis with correlated non-gaussian parameters using independent component analysis. In *DAC '06*, 2006.

[7] K. Chopra, B. Zhai, D. Blaauw, and D. Sylvester. A new statistical max operation for propagating skewness in statistical timing analysis. In *ICCAD '06*, 2006.

[8] T. Kirkpatrick and N. Clark. Pert as an aid to logic design. *IBM J. Res. Develop*, 1966.

[9] M. Berkelaar. Statistical delay calculation, a linear time method. In *Proc. TAU Int. Workshop Timing '97*, 1997.

[10] M. Imai, T. Sato, N. Nakayama, and K. Masu. Non-parametric statistical static timing analysis: An ssta framework for arbitrary distribution. In *DAC '08*, 2008.

[11] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys*, 1953.

[12] D. Azuma, S. Tsukiyama, and M. Fukui. Approximating the maximum of gaussians by a gaussian mixture model for statistical designs. In *ECCTD '17*, 2017.

[13] C. Visweswariah, J. Wang, M. Xia, D. Gangadharan, V. Tamhankar, and H. Hsieh. First-order incremental block-based statistical timing analysis. *IEEE TCAD*, 2006.

[14] C. S. Amin, R. Aitken, A. Agarwal, and I. L. Markov. Statistical static timing analysis: How simple can we get? In *DAC '05*, 2005.

[15] M. Imai, T. Sato, N. Nakayama, and K. Masu. Non-parametric statistical static timing analysis: An ssta framework for arbitrary distribution. In *DAC '08*, 2008.

[16] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1993.

[17] K. P. Murphy. *Probabilistic Machine Learning: An Introduction*. The MIT Press, 2022.

[18] J. D. Kelleher, B. M. Namee, and A. D'Arcy. *Fundamentals of Machine Learning for Predective Data Analytics*. The MIT Press, 2015.

[19] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, 2015.

[20] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[21] K. Yu and R. A. Moyeed. Bayesian quantile regression. *Stat. Probab. Lett*, 2001.

[22] V. Champac and J. G. Gervacio. *Timing Performance of Nanometer Digital Circuits Under Process Variations*. Springer International Publishing, 2018.

[23] D. Titterington, A. Smith, and U. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.