---

**Proceedings Paper:**
Zhou, J. orcid.org/0009-0009-6317-6373, Xia, H. orcid.org/0009-0007-8115-1693, Xing, W. orcid.org/0000-0002-3177-8478 et al. (3 more authors) (2025) LVFGen: Efficient Liberty Variation Format (LVF) generation using variational analysis and active learning. In: Posser, G. and Held, S., (eds.) ISPD '25: Proceedings of the 2025 International Symposium on Physical Design. ISPD '25: International Symposium on Physical Design, 16-19 Mar 2025, Austin, Texas. ACM , pp. 182-190. ISBN 9798400712937/25/03

https://doi.org/10.1145/3698364.3705359

---

# LVFGen: Efficient Liberty Variation Format (LVF) Generation Using Variational Analysis and Active Learning


Junzhuo Zhou*
junzhuo22@g.ucla.edu
University of California, Los Angeles, United States

Haoxuan Xia
hxia0118@g.ucla.edu
University of California, Los Angeles, United States

Wei Xing†
w.xing@sheffield.ac.uk
The University of Sheffield, Sheffield, United Kingdom

Ting-Jung Lin*
tlin@idt.eitech.edu.cn
Ningbo Institute of Digital Twin, Eastern Institute of Technology, Ningbo, China
BTD Inc, Ningbo, China

Li Huang
lhuang03@btd.tech
Engineering Research Center of Chiplet Design and Manufacturing of Zhejiang Province, Ningbo, China

Lei He†
lhe@ee.ucla.edu
University of California, Los Angeles, United States



## Abstract

As transistor dimensions shrink, process variations significantly impact circuit performance, signifying the need for accurate statistical circuit analysis. In digital circuit timing analysis, the Liberty Variation Format (LVF) has emerged as an industrial leading representation of timing distributions in cell libraries at 22 nm and below. However, LVF characterization relies on the Monte Carlo (MC) method, which requires excessive SPICE simulations of cells with process variations. Similar challenges also exist for uncertainty propagation and quantification in chip manufacturing and the broader scientific communities. To resolve this foundational challenge, this paper presents LVFGen, a novel method that reduces the simulation costs of MC while generate high-accuracy LVF library. LVFGen utilizes an active learning strategy based on variational analysis to identify process variation samples that impact timing distributions more significantly. Compared to the state-of-the-art Quasi-MC method, LVFGen demonstrates an overall 2.27× speedup in LVF library generation within an accuracy level of 5k-sample MC and a 4.06× speedup within a 100k-sample MC accuracy.



## CCS Concepts

• **Mathematics of computing** → **Stochastic processes**; • **Hardware** → **Statistical timing analysis**; *Modeling and parameter extraction.*

## Keywords

Statistical library generation, Yield, Active learning, Uncertainty quantification, LVF

**ACM Reference Format:**
Junzhuo Zhou, Ting-Jung Lin, Haoxuan Xia, Li Huang, Wei Xing, and Lei He. 2025. LVFGen: Efficient Liberty Variation Format (LVF) Generation Using Variational Analysis and Active Learning. In *Proceedings of the 2025 International Symposium on Physical Design (ISPD '25), March 16–19, 2025, Austin, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3698364.3705359


*Both authors contributed equally to this research.
†Corresponding authors.




*ISPD '25, Austin, TX, USA*



ACM ISBN 979-8-4007-1293-7/25/03
https://doi.org/10.1145/3698364.3705359


## 1 Introduction

The ongoing scale-down of devices significantly amplifies the impacts of process variations on circuit performance. To account for increasing manufacturing uncertainties, a global derate value is typically added in corner-based digital circuits' Static Timing Analysis (STA) techniques. However, this coarse granularity results in excessive pessimism and suboptimal designs, which severely limit circuit performance. To address these challenges, more effective On-Chip Variation (OCV) characterizations have gained increasing attention [1–3].

There have been several OCV formats which represent each timing entry as a probability distribution rather than a deterministic value. The formats are applied in Statistical Static Timing Analysis (SSTA). Relying on the distribution profiles of cells along critical paths, SSTA accurately evaluates chip timing performance during the sign-off stage. Liberty Variation Format (LVF) is the latest industry leading OCV representation in standard cell libraries at 22 nm and below [4].

While the conventional non-linear delay model (NLDM) measures deterministic timing values for each timing arc, LVF extends every measurements into a distribution and thus delivers a much higher accuracy. Modern moment-based LVF captures details in timing distributions by incorporating mean-shift, standard deviation, and skewness [5]. This comes at the costs of much longer runtime of library generation, which primarily relies on the Monte Carlo (MC) method. MC is considered the gold standard for statistical circuit analysis under process variations [6]. It provides trustworthy approximations but requires evaluating excessive process variation samples using SPICE simulations. As a vast number of cells and arcs need to be evaluated across multiple corners, millions of SPICE simulations have become a bottleneck in cell library generation, limiting large-scale applications of LVF [3].

The quasi-Monte Carlo (QMC) method boasts faster convergence and higher efficiency for statistical circuit analysis [6]. However, generating uniformly distributed QMC samples is challenging in sufficiently large dimensions [7]. A practical definition of a high-dimensional problem is when constructing equidistributed QMC points becomes problematic.

Traditional optimization techniques for LVF library generation have attempted to improve efficiency by employing machine learning for LVF data prediction [3]. Incomplete LVF data acquired by MC is used as training sets to facilitate data prediction for the remaining distributions. For example, a delay table indexed by input slews and output loads contains correlated delay distributions for different slew-load pairs. [8] addressed intra-table correlations and proposed using MC for some distributions while predicting others with machine learning. Similar correlations were observed in the same cell type with different driving strengths or across process, voltage, and temperature (PVT) corners [9]. Although these studies improved the characterization efficiency through partial data prediction, no mature method has aimed to minimize the simulation costs of a single MC for LVF library generation.

Previous work has explored the possibility of speeding up MC by importance sampling in statistical timing analysis [10–12]. However, the methods were primarily effective in finding the boundary of yield rather than the entire timing distributions required by LVF. Besides, they did not consider the impact of uncertainty quantification (UQ). UQ analysis is an essential practice that facilitates quantitative understanding of uncertainty in a model. It provides the information for predicting possibilities, inferring the outcome with the highest probability, and calibrating the model. In a broader community of UQ, it is possible to accelerate the estimation of the distribution function using active learning [13]. Despite that they can deal with arbitrarily targeted distribution, such flexibility comes with a higher computational costs. This hinders its applications in LVF that focuses on the first three moments.

This paper presents LVFGen, a novel method that addresses the above challenges by significantly reducing the simulation costs while generating highly accurate LVF library. LVFGen utilizes variational analysis and active learning to identify process variation samples with larger impacts on delay and transition distributions. Compared to MC and QMC, LVFGen achieves the same accuracy with much fewer SPICE simulations. We summarize our key contributions as follows.

- We apply variational analysis to derive a tractable analysis of the uncertainty propagation and an acquisition function for downstream applications like active learning.
- We introduce a dimension pruning method based on theoretical circuit analysis, enabling the surrogate model to achieve more accurate predictions of timing performance with uncertainties.
- LVFGen demonstrates a 2.27× overall speedup in LVF library generation with an accuracy level of 5k MC and a 4.06× speedup with a 100k-MC accuracy, compared to QMC.

To the best of our knowledge, this is the first in-depth study diving into the acceleration of characterizing a specific probability density function (PDF) by its statistical moments using a rigorous mathematical analysis tool. The key ideas of this work also benefit a broader community of UQ instead of just the EDA community.

The rest of the paper is organized as follows. Section 2 presents preliminaries and problem formulation. Section 3 introduces the proposed algorithm. Section 4 presents experimental results and Section 5 concludes the paper.

# 2 Preliminaries

## 2.1 Problem Formulation

Statistical cell library generation is used to obtain varied circuit timing performance considering process variations. This process typically involves two steps to obtain an accurate estimation of timing performance.

The first step is MC simulation, where the sample set with numerous process variation samples $\mathcal{X} = \{\boldsymbol{x_n} | \ n = 1, 2, ..., N\}$ is generated as input (with other circuit parameters fixed) for SPICE simulations. The timing analysis is obtained from the simulated waveform. This procedure can be represented as follows:

$$\boldsymbol{y_n} = f(\boldsymbol{x_n}), \tag{1}$$

where $\boldsymbol{x_n} \in \mathbb{R}^d$ represents the process variation samples from the $d$-dimension variation space. $\boldsymbol{y_n} = [y_n^{(1)}, \ y_n^{(2)}, \dots, \ y_n^{(t)}]^T$ is the $t$-dimensional results vector. Each $y_n^{(i)}$ represents one deterministic timing performance, which can be one of the delay, transition, and constraint (in sequential circuits) values.

The second step involves fitting the measured timing values into a specific statistical model. LVF employs three statistical moments, mean $\mu$, standard deviation $\sigma$, and skewness $\gamma$, to depict the timing distributions. With the collected timing result set $\mathcal{Y} = \{\boldsymbol{y_n} | \ n = 1, 2, ..., N\}$ from simulations of the first step, the three moments of the $i$-th timing performance are estimated by:

$$\begin{aligned} \mathcal{Y}^{(i)} &= \{y_n^{(i)} | \ n = 1, 2, ..., N\}, \\ \hat{\mu} &= \mathrm{E}[\mathcal{Y}^{(i)}], \\ \hat{\sigma} &= \mathrm{E}[(\mathcal{Y}^{(i)} - \mathrm{E}[\mathcal{Y}^{(i)}])^2]^{1/2}, \\ \hat{\gamma} &= \mathrm{E}[(\mathcal{Y}^{(i)} - \mathrm{E}[\mathcal{Y}^{(i)}])^3]^{1/3}. \end{aligned} \tag{2}$$

The skew-normal (SN) distribution generalizes the normal distribution with skewness. There exists a bijection between statistical moments and SN parameters: location $\xi$, scale $\omega$, and shape $\alpha$ [14]. In most applications, the statistical moments of LVF define an SN distribution whose PDF is:

$$f_{\mathrm{LVF}}(x|\mu, \sigma, \gamma) = f_{\mathrm{SN}}(x|\xi, \omega, \alpha) = \frac{2}{\omega} \phi(\frac{x-\xi}{\omega})\Phi(\alpha\frac{x-\xi}{\omega}), \tag{3}$$

where $\phi$ is the PDF of normal distribution, and $\Phi$ is the cumulative distribution function (CDF) of normal distribution.

The Central Limit Theorem (CLT) plays a crucial role in MC simulation, implying that the estimation error tends to zero as the number of samples increases with a convergence rate of $O(1/\sqrt{N})$ [15]. To reduce the number of SPICE simulations, we formulate moment-based statistical characterization as a UQ optimization problem. For the $i$-th timing performance, the goal is to minimize the distance between the estimated timing distribution and the real distribution. That is,

$$\begin{aligned} &\text{minimize} \ \ \mathcal{L}^{(i)} \\ &\text{subject to} \ \ \mathcal{L}^{(i)} = D[LVF(\hat{\mu}, \hat{\sigma}^2, \hat{\gamma}^3) | LVF(\mu, \sigma^2, \gamma^3)], \end{aligned} \tag{4}$$

where $\mathcal{L}^{(i)}$ is the objective function for $i$-th timing performance, $LVF(\hat{\mu}, \hat{\sigma}^2, \hat{\gamma}^3)$ is the timing distribution given estimated moments, $LVF(\mu, \sigma^2, \gamma^3)$ is the (unknown) real distribution, and $D[\cdot|\cdot]$ is the distance between two distributions, which is typically a divergence measure.

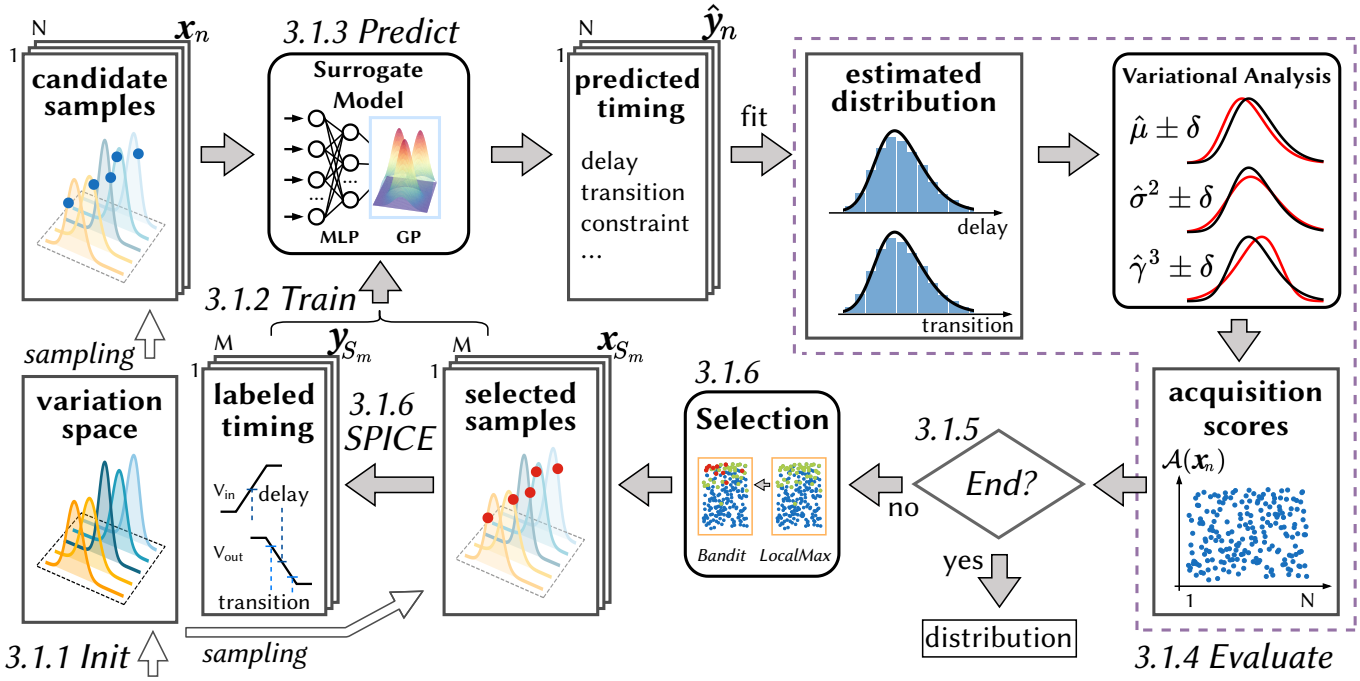



**Figure 1: Overview of our algorithm, a general active learning framework [13] application for LVF generation.**

## 2.2 Active Learning

Active learning [16] is a subfield of machine learning. By actively selecting data for learning, it can yield superior performance compared to merely receiving static data points during training. Active learning often achieves heightened accuracy with fewer labeled instances. It finds widespread application in scenarios where data abundance contrasts with label scarcity. The primary framework in active learning is uncertainty sampling. This strategy aims to diminish model variance by incorporating instances with the highest uncertainty regarding their labels into the training set during subsequent iterations of model refinement.

## 2.3 Gaussian Process

The main feature of Gaussian process (GP) is that it can predict both the mean function $\mu(\boldsymbol{x})$ of the objective $f(\boldsymbol{x})$ and the covariance matrix $k(\boldsymbol{x}, \boldsymbol{x'})$ simultaneously [17]:

$$\begin{aligned} \mu(\boldsymbol{x}) &= \mathrm{E}[f(\boldsymbol{x})], \\ k(\boldsymbol{x}, \boldsymbol{x'}) &= \mathrm{E}[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x'}) - \mu(\boldsymbol{x'}))]. \end{aligned} \tag{5}$$

When provided with input $\boldsymbol{x}$, the mean function $\mu(\boldsymbol{x})$ serves to estimate the objective value. The prevalent expression for the covariance function $k(\boldsymbol{x}, \boldsymbol{x'})$ is:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \lambda^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x'})^{\mathrm{T}} \Lambda (\boldsymbol{x} - \boldsymbol{x'})\right), \tag{6}$$

where $\Lambda = diag({\lambda_1}^{-2}, {\lambda_2}^{-2}, ..., {\lambda_d}^{-2})$ represents the square inverse singular matrix of covariance matrix $k(\boldsymbol{x}, \boldsymbol{x'})$. In our experiment, the input features comprise process variation samples, while the output objectives consist of timing performance vector with propagation delay, transition, and constraint if applicable:

$$\hat{\boldsymbol{y}}_n = g(\boldsymbol{x}_n) \sim \mathcal{N}(\mu(\boldsymbol{x}_n), v(\boldsymbol{x}_n)). \tag{7}$$

Here $v(\boldsymbol{x}_n)$ represents the predicted variance of timing vector. GP typically serves as a surrogate model adept at offering uncertainty values. It provides direct insights into the process of active learning, while the underlying real model operates as a black-box function.

However, GP has its limitations in high-dimensional and non-stationary functions [18]. This suggests that it is hard for GP to predict accurately with larger problem scales or non-linear circuit behaviors.

# 3 Proposed Algorithm

This section presents LVFGen, a statistical cell library generation algorithm that employs variational analysis to estimate the LVF timing distributions. LVFGen realizes co-optimization by selecting the effective process variation samples, whose corresponding timing results enable the surrogate model to more accurately estimate the entire distributions of delay and transition for specific circuits. The illustration in Figure 1 is assumed for combinational cells with delay and transition time. The framework is similar for sequential cells that have setup or hold time.

## 3.1 Procedures Framework

As depicted in Figure 1, LVFGen involves an iteration, each consisting of two steps. The first step labels the process variation samples with timing results using SPICE simulations, and the second step determines the most effective samples using a surrogate model combined with variational analysis. The basic procedures of the

LVFGen algorithm are listed as follows. More details are discussed in Sections 3.2-3.4.

*3.1.1 Initialization (set $k = 0$).*

- Create a large process variation sample set $\mathcal{X} = \{\boldsymbol{x}_n | \ n = 1, 2, ..., N\}$, to include all the candidate samples. Use quasi-random sequence to populate the entire process variation space.
- Take the first $M$ samples to form the initial sample set $\mathcal{X}_0 = \{\boldsymbol{x}_n | \ n = 1, 2, ..., M\}, M \ll N$.
- Based on SPICE simulation results, label the initial sample set $\mathcal{X}_0$ with timings to obtain $\mathcal{Y}_0 = f(\mathcal{X}_0) = \{f(\boldsymbol{x}_n) | \ n = 1, 2, ..., M\}$.
- Calculate the mean $\mu_0$ and standard deviation $\sigma_0$ of $\mathcal{Y}_0$ for each timing metric. Obtain the normalized initial timing performance $\mathcal{Y}_0' = (\mathcal{Y}_0 - \mu_0)/\sigma_0$.

*3.1.2 Training the Surrogate Model.*

- The sample sets $\mathcal{X}_0$ to $\mathcal{X}_k$ have been labeled with SPICE simulation results. Incrementally train the surrogate model with the labeled sets $\{(\mathcal{X}_0, \mathcal{Y}_0'), ..., (\mathcal{X}_k, \mathcal{Y}_k')\}$.

*3.1.3 Timing Prediction by Surrogate Model.*

- Use the surrogate model to predict the timing of all the candidate process variation samples in $\mathcal{X}$ and obtain $\hat{\mathcal{Y}}' = \{\hat{\boldsymbol{y}}_n' | \ \hat{\boldsymbol{y}}_n' = g(\boldsymbol{x}_n), n = 1, 2, ..., N\}$.

*3.1.4 Candidate Evaluation.*

- Evaluate the acquisition score $\mathcal{A}(\boldsymbol{x}_n)$ for each candidate sample in $\mathcal{X}$, which will be used for sample selection in step 3.1.6.

*3.1.5 Termination Criterion Check.*

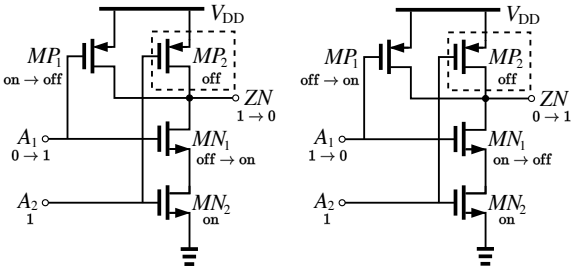
- If $k > k_{max}$ or $max\{\mathcal{A}(\boldsymbol{x}_n) | \ n = 1, 2, ..., N\} < \mathcal{A}_{min}$, terminate with the distribution of $\{\hat{\mathcal{Y}}' \times \sigma_0 + \mu_0\}$. The final distribution is determined by the timing predictions $\hat{\mathcal{Y}}'$ based on the surrogate model trained by $M \times (k+1)$ selected process variation samples.
- Else, proceed to step 3.1.6.

*3.1.6 Sample Selection & Labeling by Simulation (set $k = k + 1$).*

- Select another $M$ effective samples from $\mathcal{X}$ to form the $k$-th sample set, $\mathcal{X}_k = \{\boldsymbol{x}_n | \ n = S_1, S_2, ..., S_M\}$.
- Label the new process variation samples with SPICE simulation results to obtain $\mathcal{Y}_k = f(\mathcal{X}_k) = \{f(\boldsymbol{x}_n) | \ n = S_1, S_2, ..., S_M\}$.
- Normalize the timing performance by $\mathcal{Y}_k' = (\mathcal{Y}_k - \mu_0)/\sigma_0$.
- Back to step 3.1.2

## 3.2 Dimension Pruning

LVFGen employs a surrogate model to predict the circuit's timing performance from process variables. However, in standard cell SPICE simulations, the numerous devices and extensive process parameters result in excessively large dimensions. When GP is used directly as the surrogate model, it will be affected by the curse of dimensionality, leading to inaccurate predictions. Fortunately, the effective dimension in stochastic circuit problems can be much smaller than the intrinsic dimension of the circuit. Take a 2-input NAND circuit in Figure 2 as an instance. When the input $A_2$ stays high and $A_1$ goes rising or falling, the transistor $MP_2$ always stays off, and the process variations within $MP_2$ only weakly impact timing performance by influencing the drain parasitic capacitance on node $ZN$. We denote the inactive transistors in similar circumstances as *dark transistors*. Besides, conducting transistors may have some less significant process variations.




**(a) $A_1$ rising, $ZN$ falling** **(b) $A_1$ falling, $ZN$ rising**

**Figure 2: Dimension pruning for NAND2**

The above phenomena become more significant in large-scale circuits with more dark transistors. This implies that we can merge and prune the less significant dimensions to reduce the effective dimensionality, and thus mitigate the curse of dimensionality. In LVFGen, a multi-layer perceptron (MLP) with a fully connected activation layer is employed to perform dimension pruning. It can recognize both linear and non-linear relationships from the process variation space (up to 200 dimensions) to the pruned variable space (fewer than 20 dimensions) during the gradient descent phase of training. The pruned variables are then sent to the GP model to predict multiple timing performances with uncertainty. Now, the surrogate model of LVFGen can be presented as follows:

$$\begin{aligned} \hat{\boldsymbol{y}}_n &= g(\boldsymbol{x}_n) = g_2(g_1(\boldsymbol{x}_n)) \sim \mathcal{N}(\mu(\boldsymbol{x}_n), v(\boldsymbol{x}_n)), \\ \hat{y}_n^{(i)} &\sim \mathcal{N}(\mu^{(i)}(\boldsymbol{x}_n), v^{(i)}(\boldsymbol{x}_n)). \end{aligned} \tag{8}$$

Here $g_1 : \mathbb{R}^d \mapsto \mathbb{R}^p$ and $g_2 : \mathbb{R}^p \mapsto \mathbb{R}^t$ represents the MLP and GP transformation. $\mathbb{R}^d$ is process variation space, $\mathbb{R}^t$ is timing performance space, $\mathbb{R}^p$ is pruned variable space. The setting of pruned variables dimension $p$ should consider both the prior circuit scale and the performance of the GP model in practice. Each predicted timing performance within $\hat{\boldsymbol{y}}_n$ follows the Gaussian distribution with mean $\mu^{(i)}(\boldsymbol{x}_n)$ and variance $v^{(i)}(\boldsymbol{x}_n)$.

## 3.3 Acquisition by Variational Analysis

Recall that for $i$-th timing performance, our goal is to minimize $\mathcal{L}^{(i)}$ in Eq. (4). However, it is impossible to obtain the real distribution in practice, and the direct computation is infeasible. Thus we need to define an acquisition function to guide our optimization. The strategy of LVFGen aims to diminish the uncertainty of objective function $\mathcal{L}^{(i)}$, in the approach of finding and labeling the candidate samples who contribute the most to the uncertainty. The acquisition score provides an evaluation of the contribution, which can be formalized as follows:

$$\mathcal{A}^{(i)}(\boldsymbol{x}_n) = \mathrm{Var}[\mathcal{L}^{(i)}]\mathrm{C}_{\boldsymbol{x}_n}. \tag{9}$$

Here $\mathcal{A}^{(i)}(\boldsymbol{x}_n)$ is the acquisition score of the candidate sample $\boldsymbol{x}_n$, and $\mathrm{Var}[\mathcal{L}^{(i)}]\mathrm{C}_{\boldsymbol{x}_n}$ represents the contribution of $\boldsymbol{x}_n$ to the variance of objective function $\mathcal{L}^{(i)}$. Next, we will employ the chain rule to

express the propagation of gradient and uncertainty. **Notice** that for clarify, we will leave out all $^{(i)}$ flags on statistical moments within this section.

*3.3.1 Uncertainty Propagation along Moments.* Given the predicted timing set by the surrogate model $\{\hat{\boldsymbol{y}}_n | \hat{\boldsymbol{y}}_n = \hat{\boldsymbol{y}}'_n \times \sigma_0 + \mu_0, n = 1, 2, ..., N\}$, we want to express expectations and variances of the three statistical moments defined in Eq. (2). Since each $\hat{y}_n$ holds an equal probability, we can rewrite Eq. (2) as follows:

$$\begin{aligned} \hat{\mu} &= \frac{\sum_{n=1}^{N} \hat{y}_n^{(i)}}{N}, \\ \hat{\sigma}^2 &= \frac{\sum_{n=1}^{N} (\hat{y}_n^{(i)} - \hat{\mu})^2}{N}, \\ \hat{\gamma}^3 &= \frac{\sum_{n=1}^{N} (\hat{y}_n^{(i)} - \hat{\mu})^3}{N}. \end{aligned} \tag{10}$$

Begin with the analysis of $\hat{\mu}$, the $\sum_{n=1}^{N} \hat{y}_n$ term is a sum of normally distributed random variables. Thus, we can directly express its expectation and variance as:

$$\begin{aligned} \mathrm{E}[\hat{\mu}] &= \frac{\sum_{n=1}^{N} \mu^{(i)}(\boldsymbol{x_n})}{N}, \\ \mathrm{Var}[\hat{\mu}] &= \frac{\sum_{n=1}^{N} v^{(i)}(\boldsymbol{x_n})}{N}. \end{aligned} \tag{11}$$

For $\hat{\sigma}$ and $\hat{\gamma}$, we can find that they have a common $(\hat{y}_n^{(i)} - \hat{\mu})$ term, which is also a normally distributed random variable essentially:

$$\begin{aligned} (\hat{y}_n^{(i)} - \hat{\mu}) &\sim \mathcal{N}(\mu'_n, v'_n), \\ \mu'_n &= \mu^{(i)}(\boldsymbol{x_n}) - \mathrm{E}[\hat{\mu}], \\ v'_n &= v^{(i)}(\boldsymbol{x_n}) + \mathrm{Var}[\hat{\mu}]. \end{aligned} \tag{12}$$

With the above knowledge, we can have the expressions for $\hat{\sigma}^2$:

$$\begin{aligned} \mathrm{E}[\hat{\sigma}^2] &= \frac{\sum_{n=1}^{N} \mathrm{E}[(\hat{y}_n^{(i)} - \hat{\mu})^2]}{N} = \frac{\sum_{n=1}^{N} (\mu'^2_n + v'_n)}{N}, \\ \mathrm{Var}[\hat{\sigma}^2] &= \frac{\sum_{n=1}^{N} \mathrm{Var}[(\hat{y}_n^{(i)} - \hat{\mu})^2]}{N} = \frac{\sum_{n=1}^{N} (4\mu'^2_n v'_n + 2v'^2_n)}{N}. \end{aligned} \tag{13}$$

and the the expressions for $\hat{\gamma}^3$,

$$\begin{aligned} \mathrm{E}[\hat{\gamma}^3] &= \frac{\sum_{n=1}^{N} \mathrm{E}[(\hat{y}_n^{(i)} - \hat{\mu})^3]}{N} = \frac{\sum_{n=1}^{N} (\mu'^3_n + 3\mu'_n v'_n)}{N}, \\ \mathrm{Var}[\hat{\gamma}^3] &= \frac{\sum_{n=1}^{N} \mathrm{Var}[(\hat{y}_n^{(i)} - \hat{\mu})^3]}{N} = \frac{\sum_{n=1}^{N} (9\mu'^4_n v'_n + 36\mu'^2_n v'^2_n + 15v'^3_n)}{N}. \end{aligned} \tag{14}$$

We have forwardly expressed the expectations and variances for three statistical moments in terms of $\boldsymbol{x_n}$, with which we can easily get the gradients $\nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\mu}]$, $\nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\sigma}^2]$, and $\nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\gamma}^3]$ using computation tools like PyTorch. Finally, we express the contribution from each $\boldsymbol{x_n}$ to three statistical moments' uncertainty:

$$\begin{aligned} \mathrm{Var}[\hat{\mu}] \mathrm{C}_{\boldsymbol{x_n}} &= \nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\mu}] \cdot v(\boldsymbol{x_n}), \\ \mathrm{Var}[\hat{\sigma}^2] \mathrm{C}_{\boldsymbol{x_n}} &= \nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\sigma}^2] \cdot v(\boldsymbol{x_n}), \\ \mathrm{Var}[\hat{\gamma}^3] \mathrm{C}_{\boldsymbol{x_n}} &= \nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\gamma}^3] \cdot v(\boldsymbol{x_n}). \end{aligned} \tag{15}$$

And the estimated origin distribution statistical moments for current stage are as follows:

$$\begin{aligned} \hat{\mu}_k &= \sigma_0 \mathrm{E}[\hat{\mu}] + \mu_0, \\ \hat{\sigma}_k^2 &= \sigma_0^2 \mathrm{E}[\hat{\sigma}^2], \\ \hat{\gamma}_k^3 &= \frac{\sigma_0^2 \mathrm{E}[\hat{\gamma}^3]}{2}. \end{aligned} \tag{16}$$

*3.3.2 Variational Analysis for Objective Function.* Now we need to quantify the three statistical moments' uncertainty that influences our objective function to conduct active learning. Although direct computation is infeasible, we luckily find a way to approximately obtain the derivative using the property of distance and variational analysis. Take $\hat{\sigma}^2$ as an example:

$$\begin{aligned} &\nabla_{\hat{\sigma}^2} \mathcal{L}^{(i)} \\ &= \lim_{\delta \to 0} \frac{1}{\delta} \Big( D[LVF(\mathrm{E}[\hat{\mu}], \mathrm{E}[\hat{\sigma}^2] + \delta, \mathrm{E}[\hat{\gamma}^3]) | LVF(\mu, \sigma^2, \gamma^3)] \\ &\qquad - D[LVF(\mathrm{E}[\hat{\mu}], \mathrm{E}[\hat{\sigma}^2], \mathrm{E}[\hat{\gamma}^3]) | LVF(\mu, \sigma^2, \gamma^3)] \Big) \\ &\approx \lim_{\delta \to 0} \frac{D[LVF(\mathrm{E}[\hat{\mu}], \mathrm{E}[\hat{\sigma}^2] + \delta, \mathrm{E}[\hat{\gamma}^3]) | LVF(\mathrm{E}[\hat{\mu}], \mathrm{E}[\hat{\sigma}^2], \mathrm{E}[\hat{\gamma}^3])]}{\delta}. \end{aligned} \tag{17}$$

Here we apply the perturbation theory into variational analysis, avoiding to directly compute the $\mathcal{L}$. Then $\nabla_{\hat{\mu}} \mathcal{L}$ and $\nabla_{\hat{\gamma}^3} \mathcal{L}$ can be obtained using the same approach as Eq. (17).

*3.3.3 The Last Step.* We now assemble the variational analysis to derive a total variance change caused by each new samples as

$$\begin{aligned} &\mathrm{Var}[\mathcal{L}^{(i)}] \\ &= \left( \nabla_{\hat{\mu}} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\mu}] + \left( \nabla_{\hat{\sigma}^2} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\sigma}^2] + \left( \nabla_{\hat{\gamma}^3} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\gamma}^3] \\ &+ 2 \nabla_{\hat{\mu}} \mathcal{L} \cdot \nabla_{\hat{\sigma}^2} \mathcal{L} \cdot \mathrm{Cov}[\hat{\mu}, \hat{\sigma}^2] \\ &+ 2 \nabla_{\hat{\mu}} \mathcal{L} \cdot \nabla_{\hat{\gamma}^3} \mathcal{L} \cdot \mathrm{Cov}[\hat{\mu}, \hat{\gamma}^3] \\ &+ 2 \nabla_{\hat{\sigma}^2} \mathcal{L} \cdot \nabla_{\hat{\gamma}^3} \mathcal{L} \cdot \mathrm{Cov}[\hat{\sigma}^2, \hat{\gamma}^3] \\ &= \left( \nabla_{\hat{\mu}} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\mu}] + \left( \nabla_{\hat{\sigma}^2} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\sigma}^2] + \left( \nabla_{\hat{\gamma}^3} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\gamma}^3]. \end{aligned} \tag{18}$$

Here the co-variances between statistical moments are zeros. Combining the Eqs. (9), (15), and (18), we finally obtain the acquisition score of each candidate samples to $i$-th timing performance:

$$\begin{aligned} \mathcal{A}^{(i)}(\boldsymbol{x_n}) &= \mathrm{Var}[\mathcal{L}^{(i)}] \mathrm{C}_{\boldsymbol{x_n}} \\ &= \left( \nabla_{\hat{\mu}} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\mu}] \mathrm{C}_{\boldsymbol{x_n}} + \left( \nabla_{\hat{\sigma}^2} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\sigma}^2] \mathrm{C}_{\boldsymbol{x_n}} \\ &+ \left( \nabla_{\hat{\gamma}^3} \mathcal{L}^{(i)} \right)^2 \cdot \mathrm{Var}[\hat{\gamma}^3] \mathrm{C}_{\boldsymbol{x_n}} \\ &= \left( \nabla_{\hat{\mu}} \mathcal{L}^{(i)} \right)^2 \cdot \nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\mu}] \cdot v(\boldsymbol{x_n}) \\ &+ \left( \nabla_{\hat{\sigma}^2} \mathcal{L}^{(i)} \right)^2 \cdot \nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\sigma}^2] \cdot v(\boldsymbol{x_n}) \\ &+ \left( \nabla_{\hat{\gamma}^3} \mathcal{L}^{(i)} \right)^2 \cdot \nabla_{v(\boldsymbol{x_n})} \mathrm{Var}[\hat{\gamma}^3] \cdot v(\boldsymbol{x_n}). \end{aligned} \tag{19}$$

### 3.4 Multi-timing Co-optimization

Having an acquisition function that only addresses single-timing performance is insufficient, as one SPICE simulation yields multiple timing results (such as transition and delay) simultaneously. Therefore, it is crucial for our algorithm to support multi-timing co-optimization, both in evaluating candidate acquisitions and in selecting effective samples.

*3.4.1 Acquisition Evaluation.* We use an intuitive weighting approach to evaluate the acquisition scores of candidate samples for multi-timing objective functions:

$$\mathcal{A}(\boldsymbol{x}_n) = \sum_{i=1}^{t} \left( w_i \cdot \mathcal{A}^{(i)}(\boldsymbol{x}_n) \right). \tag{20}$$

Here $\mathcal{A}(\boldsymbol{x}_n)$ is the overall acquisition scores of candidate sample $\boldsymbol{x}_n$, and $w_i$ is the weight factor for the $i$-th timing distribution.

*3.4.2 Effective Sample Selection.* A higher acquisition score means a larger contribution to the uncertainty. However, in practice, we find that the rare tail samples tend to have high acquisition scores, while the extensive body samples typically have lower acquisition scores. Hence, the sample selection strategy needs to consider the candidate samples' locations in the distribution, rather than simply basing decisions on the acquisition scores. The algorithm firstly applies a local max operation to identify the local maximum acquisition scores $\{\mathcal{A}(\boldsymbol{x}_n), n = L_1, L_2, ..., L_{M'}\}$ considering the samples' locations. Then we use normalized acquisition values as the probabilities for selecting $M$ effective samples from the local maximum samples:

$$P(\boldsymbol{x}_n) = \frac{\mathcal{A}(\boldsymbol{x}_n)}{\sum_{n=1}^{L_{M'}} \mathcal{A}(\boldsymbol{x}_n)} \quad n = L_1, L_2, ..., L_{M'}. \tag{21}$$

Finally, this multi-armed bandit problem will return the selected effective sample set for $k$-th iteration $\mathcal{X}_k = \{\boldsymbol{x}_n, n = S_1, S_2, ..., S_M\}$. The new samples set will be labeled with SPICE simulations and used to train the surrogate model for next iteration.

## 4 Experiment Results and Discussions

### 4.1 Experiment Setting

The experiments use the TSMC 22nm standard cells under 0.8V, $25°C$, and `TTGlobal_LocalMC` corner with all local variations turned on. The proposed method is implemented in PyTorch and assessed by HSPICE on Linux machines with NVIDIA 3090 GPU and 8-core Intel Xeon 6348 CPU, each machine running 8 tasks simultaneously. For other algorithms, we use pure CPU machines (as they are not designed to utilize the GPU). All the runtime reported below has been converted to that of a single thread.

We compare three algorithms: LVFGen, Sobol's QMC [19], and Random MC. Sobol's QMC is a state-of-the-art algorithm with notable enhancement in convergence for stochastic circuit problems. Random MC refers to random sampling, serving as the baseline. We report the comparisons for one standard cell in Section 4.2 and a library of 26 typical cells in 4.3. Applications of the generated LVF to SSTA are presented in Section 4.4.

We use results obtained from Random MC with 100k samples as golden. To evaluate the accuracy of the estimated distributions, we use Jensen–Shannon divergence (J-divergence) as the metric to measure the distance between the estimate and golden. J-divergence is a symmetric measure and it satisfies the triangular inequality [20], which reduces the approximation error of Eq. (17). The definition of J-divergence is:

$$\begin{aligned} D_J[LVF_1 \| LVF_2] &= D_{KL}[LVF_1 \| LVF_2] + D_{KL}[LVF_2 \| LVF_1], \\ D_{KL}[LVF_1 \| LVF_2] &= \int f_{\text{LVF1}}(x) \log\left(\frac{f_{\text{LVF1}}(x)}{f_{\text{LVF2}}(x)}\right) dx. \end{aligned} \tag{22}$$

Here $D_{KL}$ is the Kullback-Leibler divergence, and $f_{\text{LVF}}(x)$ is the PDF of LVF timing.

For ease of reading, we denote the J-divergence between golden and the estimation of Random MC of 5k samples as **5k Accuracy** With similar conventions, we also define [**10k**, **25k**, **50k**, **100k**] **Accuracy**. The algorithms are assessed under different accuracy requirements to provide a comprehensive comparison.

In experiments, we have the candidate sample set $\mathcal{X}$ with size $N = 2^{20}$. The selected sample set $\mathcal{X}_k$ has size $M = 64$ for the $k$-th iteration. The pruned variable dimension varies from 10 to 16, depending on the circuit scale. Parameters of the surrogate model are initialized randomly. SPICE simulation labels each sample with one delay and one transition. Since cell delays are more significant than transitions in timing propagation [21], we weigh the delay and transition acquisition scores with 0.7 and 0.3 respectively in Eq. (20).

### 4.2 Convergence for OR2 Cell

This experiment assesses LVFGen using a 48-dimensional OR2X2 cell. The timing arc has one input rising and the other staying low, resulting in a rising output.

Figure 3 presents the delay and transition distributions estimated by LVFGen, along with the acquisition scores for each potential candidates. The $D_J$ values are the J-divergence from golden to the estimated distribution. Figures 3a-3e show how the estimations improve over iterations. We can see the curves rapidly converge to golden along with smaller values of J-divergence. The projection plots of acquisition scores evidence that the selection strategy successfully identifies the most contributing samples from millions of candidates, with a balanced population in distribution body and tail. Additionally, we observe that the acquisition scores decrease over iterations. This indicates that the surrogate model can effectively generalize the mapping from process variations to timing performances based on previous learning, thereby reducing the acquisition scores of unlabeled samples.

Figures 4a compare the efficiency of Random MC, Sobol's, and LVFGen in estimating delay distributions. The plots are in $\log_{10} - \log_{10}$ scale, where the slopes of the linear fitting indicate the convergence rates. The $O(1/N)$ convergence rate of Random MC is aligned with the theoretical $O(1/\sqrt{N})$ convergence rate of CLT since the J-divergence is a square measure. The rapid decreasing trend of LVFGen demonstrates its strength in fast convergence.

Figure 4c shows the accuracy of LVFGen prediction reaches saturation quickly after achieving 100k accuracy level. We believe the saturation happens due to a more complex mapping from process variations to transitions, which diminishes the effectiveness of LVFGen. Considering transition's self-adaptive property in timing propagation [22, 23], the 100k accuracy with J-divergence below $10^{-4}$ only has a light impact on SSTA. Figure 3e proves that LVFGen

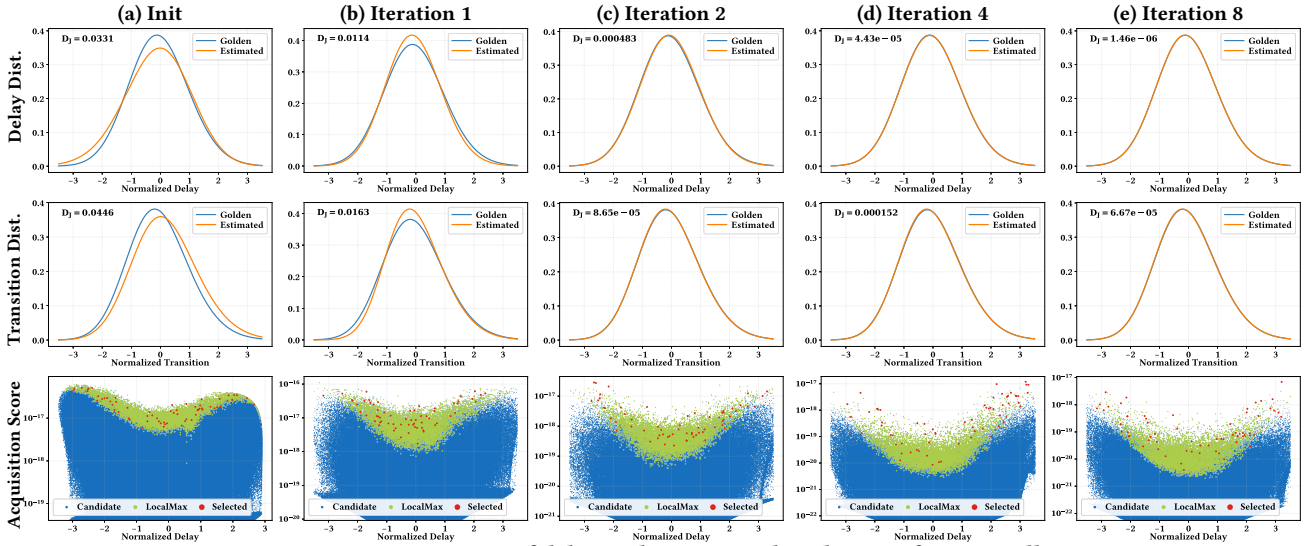



**Figure 3: Convergence of delay and transition distributions for OR2 cell.**

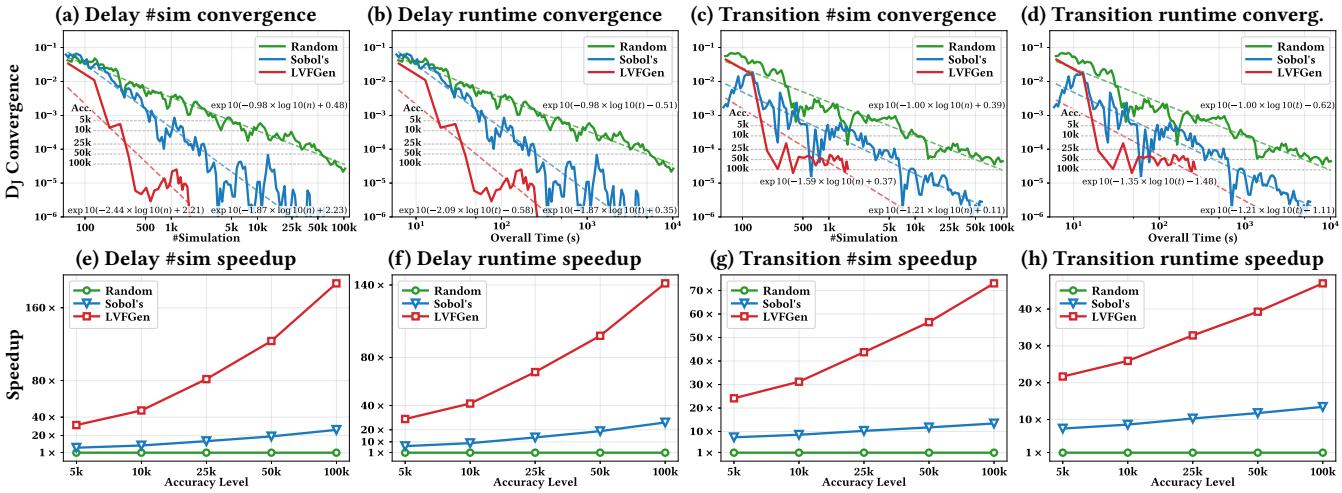



**Figure 4: Comparison of speedup between LVFGen, Random MC, and Sobol's for OR2 cell.**

already achieves sufficient accuracy in predicting transition distributions. In summary, LVFGen achieves the 100k accuracy level with only ~300 simulations, compared to the 100k simulations required by Random MC and ~5k simulations by Sobol's.

Similarly, Figures 4e and 4g plot the speedup rates of Sobol's and LVFGen in terms of the reduction in simulation times across several accuracy levels. LVFGen demonstrates a significant speedup in estimating both delay and transition distributions compared to Random MC and Sobol's. Furthermore, LVFGen proves its effectiveness with even larger improvements at higher accuracy requirements.

For fair comparisons, we include the overhead of active learning computations rather than just the simulation times. Figures 4b and 4d present the convergence comparisons with runtime. Figures 4f and 4g plot the runtime improvements.

Although the algorithm overhead degrades the efficiency of LVFGen, it still achieves significant speedup. Compared to Random MC, LVFGen saves up to 140× and 40× runtime for delay and transition estimations, the speedups for Sobol's are up to 27× and 13×. Compared to Sobol's QMC, the speedups for LVFGen are to 5× and 3×, respectively.

## 4.3 Speedup for Cell Library

To prove the generalization of LVFGen, we conduct an extensive experiment over 26 distinct standard cells, with dimensions ranging from 36 to 156. According to design complexities, we cluster those cells into three categories: low-dimension, middle-dimension, and high-dimension. The evaluated accuracy level ranges from 5k to 100k, covering the basic requirements in practice [24–27].

We assess each cell with Random MC, Sobol's QMC and LVFGen, each estimating 8×8 slow-load pairs' timing distributions of selected arcs. Table 1 summarizes the results.

We assume six variation variables for each transistor. Cells with the same function but different drive strengths may use different numbers of transistors, which results in different number of variables. However, the variations of parallel transistors tend to have a similar effect on timing. Thus, we hypothesize that the MLP can identify the similarity and merge the variables for dimension reduction. Consequently, the pruned dimensions are determined by the complexity of schematic topology rather than the number of transistors.

**Table 1: Speedup Comparison Using Standard Cell Library.**

| Cell Type* | #Transistor | Origin dim. | Pruned dim. | Delay Runtime Speedup† (×) | | | | | | | | Transition Runtime Speedup† (×) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 5k Accuracy | | 10k Accuracy | | 50k Accuracy | | 100k Accuracy | | 5k Accuracy | | 10k Accuracy | | 50k Accuracy | | 100k Accuracy | |
| | | | | Sobol's | Ours | Sobol's | Ours | Sobol's | Ours | Sobol's | Ours | Sobol's | Ours | Sobol's | Ours | Sobol's | Ours | Sobol's | Ours |
| OR2X1 | 6 | 36 | 12 | 9.23 | **34.99** | 10.95 | **46.00** | 15.91 | **85.02** | 18.54 | **110.71** | 4.36 | **10.62** | 4.60 | **11.51** | 5.30 | **12.96** | 5.68 | **13.30** |
| OR2X2 | 8 | 48 | 12 | 8.45 | **26.80** | 10.41 | **42.11** | 15.89 | **118.03** | 18.67 | **182.37** | 4.99 | **10.29** | 6.25 | **13.57** | 10.15 | **25.28** | 12.40 | **32.24** |
| NOR2X1 | 4 | 24 | 10 | 8.99 | **56.35** | 9.95 | **74.81** | 12.48 | **142.98** | 13.75 | **187.76** | 3.31 | **5.45** | 3.66 | **5.08** | **4.48** | 3.76 | **4.86** | 3.13 |
| NOR2X2 | 8 | 48 | 10 | 3.56 | **9.30** | 3.72 | **12.66** | 4.07 | **25.63** | 4.21 | **34.99** | **2.15** | 1.37 | **2.34** | 1.42 | **2.85** | 1.51 | **3.10** | 1.54 |
| AND2X1 | 6 | 36 | 10 | 26.02 | **63.02** | 33.54 | **90.60** | 60.51 | **207.56** | 78.24 | **294.38** | 10.88 | **11.30** | **11.88** | 11.23 | **14.51** | 10.10 | **15.68** | 9.34 |
| AND2X2 | 8 | 48 | 10 | 17.59 | **48.12** | 22.22 | **73.23** | 38.08 | **196.07** | 47.80 | **298.94** | 7.13 | **16.84** | 7.97 | **22.15** | 10.04 | **39.51** | 10.97 | **51.11** |
| NAND2X1 | 4 | 24 | 12 | 82.08 | **247.67** | 114.61 | **365.67** | 254.07 | **876.82** | 360.19 | **1262.33** | **19.07** | 11.11 | **25.65** | 10.49 | **49.33** | 9.37 | **64.52** | 8.34 |
| NAND2X2 | 8 | 48 | 12 | 8.86 | **20.58** | 10.20 | **28.64** | 14.11 | **62.54** | 16.22 | **87.48** | **4.89** | 1.92 | **5.57** | 2.00 | **7.46** | 2.16 | **8.43** | 2.23 |
| AOI21X1 | 6 | 36 | 16 | 62.72 | **156.12** | 95.28 | **258.62** | 250.36 | **825.74** | 378.69 | **1354.56** | 20.16 | **34.11** | 28.00 | **48.22** | 59.59 | **102.51** | 82.32 | **138.16** |
| OAI21X1 | 6 | 36 | 16 | 37.83 | **106.35** | 48.08 | **157.71** | 83.02 | **396.32** | 104.60 | **584.31** | **11.79** | 3.99 | **13.78** | 3.19 | **19.79** | 1.70 | **23.05** | 1.26 |
| **Low-dim. Avg.** | | | | 26.53 | **76.93** | 35.90 | **115.01** | 74.85 | **293.67** | 104.09 | **439.78** | 8.87 | **10.70** | 10.97 | **12.89** | 18.35 | **20.89** | 23.10 | **26.06** |
| XNOR2X1 | 12 | 72 | 16 | 30.13 | **34.35** | 34.25 | **53.61** | 44.88 | **150.67** | 50.22 | **234.26** | 12.61 | **26.34** | 14.19 | **40.22** | 18.14 | **106.02** | 19.87 | **160.88** |
| XNOR2X2 | 14 | 84 | 16 | 5.99 | **18.60** | 6.19 | **27.24** | 6.50 | **65.76** | 6.58 | **95.91** | 2.89 | **14.65** | 3.21 | **21.32** | 4.07 | **50.70** | 4.53 | **73.46** |
| XOR2X1 | 12 | 72 | 16 | 8.47 | **25.85** | 9.98 | **38.81** | 14.16 | **98.52** | 16.40 | **146.28** | 12.34 | **25.30** | 13.55 | **37.96** | 17.09 | **97.25** | 18.34 | **145.01** |
| XOR2X2 | 14 | 84 | 16 | 22.37 | **31.83** | 29.92 | **50.72** | 58.24 | **150.20** | 77.72 | **239.00** | 8.24 | **22.39** | 10.58 | **34.32** | 18.57 | **91.10** | 23.38 | **137.97** |
| AOI21X2 | 12 | 72 | 16 | 11.06 | **20.28** | 13.57 | **31.52** | 21.56 | **87.20** | 26.17 | **134.96** | **4.17** | 3.97 | **4.68** | 4.37 | **6.10** | 4.93 | **6.80** | 5.02 |
| OAI21X2 | 12 | 72 | 16 | 4.13 | **20.39** | 4.56 | **30.40** | 5.70 | **76.58** | 6.26 | **113.99** | 1.43 | **5.29** | 1.47 | **6.68** | 1.57 | **11.28** | 1.61 | **14.02** |
| **Middle-dim. Avg.** | | | | 13.69 | **25.22** | 16.41 | **38.72** | 25.17 | **104.82** | 30.56 | **160.73** | 6.95 | **16.32** | 7.95 | **24.15** | 10.92 | **60.21** | 12.42 | **89.39** |
| OR2X4 | 16 | 96 | 12 | 16.92 | **19.25** | 20.48 | **29.95** | 31.80 | **82.82** | 38.37 | **128.05** | 9.77 | **10.84** | 12.40 | **15.66** | 21.27 | **36.51** | 26.55 | **52.39** |
| NOR2X4 | 16 | 96 | 10 | 8.38 | **13.36** | 9.75 | **19.88** | 13.87 | **49.44** | 16.13 | **72.82** | 4.16 | **5.26** | 4.79 | **6.87** | 6.51 | **12.57** | 7.38 | **16.18** |
| AND2X4 | 16 | 96 | 10 | 16.66 | **16.93** | 22.55 | **27.65** | 44.73 | **85.25** | 59.73 | **137.77** | **9.53** | 8.43 | **12.80** | 12.54 | 24.54 | **30.93** | 32.11 | **45.27** |
| NAND2X4 | 16 | 96 | 12 | **2.50** | 1.96 | **2.51** | 2.30 | 2.53 | **3.21** | 2.54 | **3.65** | **1.50** | 1.32 | **1.44** | 1.34 | 1.31 | **1.36** | 1.26 | **1.36** |
| HA1X1 | 20 | 120 | 16 | **15.28** | 11.38 | **19.48** | 17.27 | 32.88 | **45.37** | 41.04 | **68.49** | **9.10** | 4.82 | **10.36** | 6.64 | **14.10** | 13.72 | 16.24 | **18.60** |
| HA1X2 | 24 | 144 | 16 | **10.12** | 6.02 | **11.53** | 9.02 | 15.96 | **22.82** | 18.52 | **33.95** | **3.63** | 2.82 | **4.20** | 4.02 | 6.11 | **9.07** | 7.18 | **12.85** |
| XNOR2X4 | 26 | 156 | 16 | **5.14** | 2.07 | **5.85** | 2.86 | **7.77** | 6.03 | **8.73** | **8.29** | 4.14 | 1.41 | **4.80** | 1.87 | **6.73** | 3.59 | **7.77** | 4.73 |
| XOR2X4 | 26 | 156 | 16 | **7.48** | 1.89 | **8.62** | 2.57 | **11.13** | 5.17 | **12.10** | 6.93 | **4.57** | 1.96 | **4.98** | 2.66 | **6.07** | 5.29 | 6.60 | **7.05** |
| AOI21X4 | 24 | 144 | 16 | **6.14** | 3.60 | **7.97** | 5.38 | **14.69** | 13.69 | 19.18 | **20.45** | **2.51** | 2.10 | **3.30** | 3.00 | 6.24 | **6.81** | 8.20 | **9.67** |
| OAI21X4 | 24 | 144 | 16 | 3.88 | **4.50** | 4.19 | **6.53** | 4.93 | **15.51** | 5.27 | **22.51** | 1.56 | **1.83** | 1.54 | **2.43** | 1.47 | **4.66** | 1.43 | **6.13** |
| **High-dim. Avg.** | | | | 9.25 | **8.10** | 11.29 | **12.34** | 18.03 | **32.93** | 22.16 | **50.29** | **5.05** | 4.08 | **6.06** | 5.70 | 9.44 | **12.45** | 11.47 | **17.42** |
| **Overall** | | | | 16.92 | **38.52** | 21.94 | **57.91** | 41.53 | **149.81** | 55.61 | **225.58** | 6.96 | **9.45** | 8.38 | **12.72** | 13.21 | **26.72** | 16.16 | **37.35** |
| **Speedup Compare to Sobol's** | | | | 1× | **2.28×** | 1× | **2.64×** | 1× | **3.61×** | 1× | **4.06×** | 1× | **1.36×** | 1× | **1.52×** | 1× | **2.02×** | 1× | **2.31×** |

* **One timing arc test for each cell type, containing 8×8 delay and transition distributions;** † **Runtime speedup compared to Random MC.**

**Table 2: Accuracy Comparison for ISCAS'89 circuits, Golden Uses 100k Accuracy for Cells.**

| Benchmark | #Gates | Mean for Critical Delay (ns) | | | Std. Dev. for Critical Delay (ns) | | |
|---|---|---|---|---|---|---|---|
| | | **MC** 100k | **Sobol's** 5k | **LVFGen** 0.7k | **MC** 100k | **Sobol's** 5k | **LVFGen** 0.7k |
| s27 | 15 | 13.031190(0‰) | 13.031190 ( 0‰) | 13.031192 (1.53E-04‰) | 0.000956(0‰) | 0.000955 ( 1.05‰) | 0.000956 ( 0‰) |
| s298 | 106 | 0.086129(0‰) | 0.086129 ( 0‰) | 0.086129 ( 0‰) | 0.007545(0‰) | 0.007545 ( 0‰) | 0.007545 ( 0‰) |
| s641 | 129 | 13.197377(0‰) | 13.197379 (1.52E-04‰) | 13.197384 (5.30E-04‰) | 0.003137(0‰) | 0.003134 (9.56E-01‰) | 0.003139 (6.38E-01‰) |
| s1196 | 519 | 13.194728(0‰) | 13.194734 (4.55E-04‰) | 13.194733 (3.79E-04‰) | 0.002978(0‰) | 0.002976 (6.72E-01‰) | 0.002979 (3.36E-01‰) |
| s15850 | 589 | 13.025476(0‰) | 13.025476 ( 0‰) | 13.025476 ( 0‰) | 0.001611(0‰) | 0.001611 ( 0‰) | 0.001613 ( 1.24‰) |
| s9234_1 | 1045 | 13.053585(0‰) | 13.053590 (3.83E-04‰) | 13.053593 (6.13E-04‰) | 0.001939(0‰) | 0.001938 (5.16E-01‰) | 0.001940 (5.16E-01‰) |
| s13207 | 1067 | 13.024756(0‰) | 13.024757 (7.68E-05‰) | 13.024757 (7.68E-05‰) | 0.001007(0‰) | 0.001006 (9.93E-01‰) | 0.001010 ( 2.98‰) |
| s5378 | 1524 | 13.132874(0‰) | 13.132888 (1.07E-03‰) | 13.132887 (9.90E-04‰) | 0.003658(0‰) | 0.003656 (5.47E-01‰) | 0.003664 ( 1.64‰) |
| MRAE* | | - | 1× , 2.67E-4‰ | 1.29× , 3.43E-4‰ | - | 1× , 0.59‰ | 1.55× , 0.92‰ |

* **Mean Relative Absolute Error.**

The runtime speedup is evaluated similarly to the previous experiment. The results in Table 1 indicate that, in almost all cases, LVFGen achieves a higher speedup than Sobol's compared to Random MC. Generally, we observe that the speedup of Sobol's is not promising as the circuit dimension increases. Although LVFGen encounters the same high-dimension issue, it is still faster than the other two algorithms.

To summarize, at a 5k accuracy level, LVFGen achieves 38.43× and 2.27× overall speedups compared to Random MC and Sobol's in predicting delay distributions, and 9.46× and 1.23× for transition distributions. LVFGen demonstrates more improvements with higher accuracy requirements. At the 100k accuracy level, LVFGen achieves 225.58× and 4.06× speedups in estimating delay distributions, 37.35× and 2.31× in estimating transition, compared to Random MC and Sobol's QMC algorithms.

## 4.4 Application to Timing Analysis

At timing sign-off, larger-scale circuits demand accurate timing distributions of cells to reduce the impacts of accumulated errors along timing propagation. In this experiment, we utilize the standard statistical timing analysis tool, PrimeTime, to assess the accuracy of LVF library characterized by LVFGen. The benchmarks circuits are selected from ISCAS'89 with scales up to 1,524 gates [28].

We first prepare three LVF libraries with the same 100k-MC accuracy level, generated by Random MC with 100k samples, Sobol's QMC with 5k samples, and LVFGen with 0.7k samples. In terms of library generation efficiency, the runtime for Sobol's QMC is ~290 hours, whereas LVFGen requires only ~80 hours.

For the SSTA procedure, the critical path is initially identified under the Golden design flow. Subsequently, we compared the path delays with three different LVF library files. We observe extremely small errors across the benchmarks shown in Table 2. The relative absolute error of LVFGen is no more than 1E-3‰ for mean delay,

and no more than 3‰ for standard deviation. The mean relative absolute error (MRAE) LVFGen achieves is remarkably close to Sobol's, with only 1.29× that of Sobol's in mean, and 1.55× in standard deviation of the critical path delay. In summary, the entire library generation of LVFGen achieves is about 3.5× faster than Sobol's QMC, providing a more-than-sufficient accuracy level in real applications.

## 5 Conclusion

This paper presents a novel method, LVFGen, which generates highly accurate LVF library with much less simulation costs compared to conventional Monte Carlo (MC) and quasi Monte Carlo (QMC) methods. LVFGen utilizes variational analysis and active learning to identify process variation samples with larger impacts on delay and transition distributions. Experimental results based on TSMC 22nm standard cells prove LVFGen's stability and efficiency in real applications. LVFGen achieves up to 225.58× and 4.06× overall speedups compared to MC and Sobol's QMC on delay estimation, 37.35× and 2.31× overall speedups compared to MC and Sobol's on transition estimation. With a real circuit and cell library, LVFGen shows a 3.5× statistical library generation speedup with competitive accuracy in SSTA evaluation. We hope the idea of LVFGen can inspire more interesting researches that adopt UQ to solve circuit problems. The further extension includes extending LVFGen with transfer learning to let one surrogate model learn the timing information from different circuits.

## Acknowledgments

The authors would like to thank Tao Bai, for assistance in the circuit path experiment.

## References

[1] Hanif Fatemi, Shahin Nazarian, and Massoud Pedram. Statistical logic cell delay analysis using a current-based model. In *Proceedings of the 43rd annual Design Automation Conference*, pages 253–256, 2006.

[2] Leronq Cheng, Jinjun Xiong, and Lei He. Non-Gaussian Statistical Timing Analysis Using Second-Order Polynomial Fitting. In *2008 Asia and South Pacific Design Automation Conference*, pages 298–303, 2008.

[3] Eunice Naswali, Namhoon Kim, and Pravin Chandran. Fast and Accurate Library Generation Leveraging Deep Learning for OCV Modelling. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 349–354. IEEE, 2021.

[4] Tan and Dyck. Advanced solutions for LVF .LIB. Siemens Digital Industries Software, 2023. https://www.eda-solutions.com/app/uploads/2023/08/Siemens-Advanced-solutions-for-LVF-.LIB-WP-82833-C1.pdf.

[5] Synopsys. Liberty Release Notes Version 2017.06. https://www.synopsys.com/cgi-bin/tapin/docsdl/Liberty_2017.06_relnotes.pdf.

[6] Amith Singhee and Rob A. Rutenbar. Why Quasi-Monte Carlo is Better Than Monte Carlo or Latin Hypercube Sampling for Statistical Circuit Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(11):1763–1776, 2010.

[7] A.B. Owen. Monte Carlo extension of quasi-Monte Carlo. In *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, volume 1, pages 571–577 vol.1, 1998.

[8] Eunice Naswali, Adalberto Claudio Quiros, and Pravin Chandran. DNNLibGen: Deep Neural Network Based Fast Library Generator. In *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 574–577. IEEE, 2019.

[9] Tianliang Ma, Zhihui Deng, Xuguang Sun, and Leilai Shao. Fast Cell Library Characterization for Design Technology Co-Optimization Based on Graph Neural Networks. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 472–477. IEEE, 2024.

[10] Jun Tao, Shupeng Sun, Xin Li, Hongzhou Liu, Kangsheng Luo, Ben Gu, and Xuan Zeng. Fast Statistical Analysis of Rare Circuit Failure Events. *Machine Learning in VLSI Computer-Aided Design*, pages 349–373, 2019.

[11] Liang Pang, Mengyun Yao, and Yifan Chai. An efficient SRAM yield analysis using scaled-sigma adaptive importance sampling. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 97–102. IEEE, 2020.

[12] Mariam Rakka, Rouwaida Kanj, and Ragheb Raad. Hybrid importance splitting importance sampling methodology for fast yield analysis of memory designs. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2020.

[13] Ziqi Wang and Marco Broccardo. A novel active learning-based Gaussian process metamodelling strategy for estimating the full probability distribution in forward UQ analysis. *Structural Safety*, 84:101937, 2020.

[14] Adelchi Azzalini and Antonella Capitanio. Statistical applications of the multivariate skew normal distribution. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):579–602, 1999.

[15] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, January 1968.

[16] Burr Settles. Active learning literature survey. 2009.

[17] Qi Sun, Tinghuan Chen, Siting Liu, Jianli Chen, Hao Yu, and Bei Yu. Correlated multi-objective multi-fidelity optimization for HLS directives design. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 27(4):1–27, 2022.

[18] Yucen Lily Li, Tim GJ Rudner, and Andrew Gordon Wilson. A study of Bayesian neural network surrogates for Bayesian optimization. *arXiv preprint arXiv:2305.20028*, 2023.

[19] I. M. Sobol'. The distribution of points in a cube and the accurate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[20] Javier E. Contreras-Reyes and Reinaldo Boris Arellano-Valle. Kullback-Leibler Divergence Measure for Multivariate Skew-Normal Distributions. *Entropy*, 14:1606–1626, 2012.

[21] P. Maurine, M. Rezzoug, N. Azemard, and D. Auvergne. Transition time modeling in deep submicron CMOS. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(11):1352–1363, 2002.

[22] D. Blaauw, V. Zolotov, S. Sundareswaran, C. Oh, and R. Panda. Slope Propagation in Static Timing Analysis. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140)*, pages 338–343, 2000.

[23] S. Nazarian, M. Pedram, E.T. Tuncer, Tao Lin, and A.H. Ajami. Modeling and Propagation of Noisy Waveforms in Static Timing Analysis. In *Design, Automation and Test in Europe*, pages 776–777 Vol. 2, 2005.

[24] Lerong Cheng, Jinjun Xiong, and Lei He. Non-Gaussian Statistical Timing Analysis Using Second-Order Polynomial Fitting. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(1):130–140, 2009.

[25] Debjit Sinha, Vasant Rao, Chaitanya Peddawad, Michael Wood, Jeffrey Hemmett, Suriya Skariah, and Patrick Williams. Statistical timing analysis considering multiple-input switching. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[26] C. Visweswariah, K. Ravindran, K. Kalafala, S.G. Walker, S. Narayan, D.K. Beece, J. Piaget, N. Venkateswaran, and J.G. Hemmett. First-Order Incremental Block-Based Statistical Timing Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2170–2180, 2006.

[27] Lizheng Zhang, W. Chen, Y. Hu, and C.C. Chen. (statistical static timing analysis with conditional linear max/min approximation and extended canonical timing model). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1183–1191, 2006.

[28] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. In *1989 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1929–1934 vol.3, 1989.