Wages: The Worst Transistor Aging Analysis for Large-scale Analog Integrated Circuits via Domain Generalization

TINGHUAN CHEN, School of Science and Engineering, The Chinese University of Hong Kong - Shenzhen, Shenzhen, China
HAO GENG, School of Information Science and Technology, ShanghaiTech University, Shanghai, China
QI SUN, College of Integrated Circuits, Zhejiang University, Hangzhou, China
SANPING WAN, HiSilicon Technologies Co., Shenzhen, China
YONGSHENG SUN, HiSilicon Technologies Co., Shenzhen, China
HUATAO YU, HiSilicon Technologies Co., Shenzhen, China
BEI YU, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China

Transistor aging leads to the deterioration of analog circuit performance over time. The worst aging degradation is used to evaluate the circuit reliability. It is extremely expensive to obtain it since several circuit stimuli need to be simulated. The worst degradation collection cost reduction brings an inaccurate training dataset when a machine learning (ML) model is used to fast perform the estimation. Motivated by the fact that there are many similar subcircuits in large-scale analog circuits, in this article we propose *Wages* to train an ML model on an inaccurate dataset for the worst aging degradation estimation via a domain generalization technique. A sampling-based method on the feature space of the transistor and its neighborhood subcircuit is developed to replace inaccurate labels. A consistent estimation for the worst degradation is enforced to update model parameters. Label updating and model updating are performed alternately to train an ML model on the inaccurate dataset. Experimental results on the very advanced 5*nm* technology node show that our Wages can significantly reduce the label collection cost with a negligible estimation error for the worst aging degradations compared to the traditional methods.

CCS Concepts: • **Hardware → Electronic design automation**; **Methodologies for EDA;**

Additional Key Words and Phrases: Analog circuits, aging, reliability, machine learning

This work is supported in part by The National Key R&D Program of China (No. 2023YFB4402900), The Research Grants Council of Hong Kong SAR (No. CUHK14208021), The National Natural Science Foundation of China (No. 62304197), HiSilicon and The Shanghai Pujiang Program (Project No. 22PJ1410400).
Authors' Contact Information: Tinghuan Chen, School of Science and Engineering, The Chinese University of Hong Kong - Shenzhen, Shenzhen, China; e-mail: chentinghuan@cuhk.edu.cn; Hao Geng, School of Information Science and Technology, ShanghaiTech University, Shanghai, China; e-mail: genghao@shanghaitech.edu.cn; Qi Sun, College of Integrated Circuits, Zhejiang University, Hangzhou, Zhejiang, China; e-mail: qisunchn@zju.edu.cn; Sanping Wan, HiSilicon Technologies Co., Shenzhen, China; e-mail: wansanping@hisilicon.com; Yongsheng Sun, HiSilicon Technologies Co., Chengdu, China; e-mail: yshsun@hisilicon.com; Huatao Yu, HiSilicon Technologies Co., Shenzhen, China; e-mail: yuhuatao@hisilicon.com; Bei Yu, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China; e-mail: byu@cse.cuhk.edu.hk.

ACM 1084-4309/2024/08-ART73
https://doi.org/10.1145/3659950

ACM Reference Format:
Tinghuan Chen, Hao Geng, Qi Sun, Sanping Wan, Yongsheng Sun, Huatao Yu, and Bei Yu. 2024. Wages: The Worst Transistor Aging Analysis for Large-scale Analog Integrated Circuits via Domain Generalization. *ACM Trans. Des. Autom. Electron. Syst.* 29, 5, Article 73 (August 2024), 23 pages. https://doi.org/10.1145/3659950

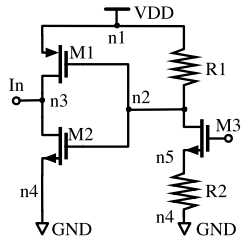
1 Introduction

Transistor aging is a major reliability issue for analog **integrated circuits (ICs)** under advanced nanometer technologies. Very small transistor sizes, combined with increasing gate-oxide electric fields, are the major sources of this problem. The performance of nanometer-scale ICs is adversely affected by transistor aging induced by **bias temperature instability (BTI)** and **hot carrier injection (HCI)** [1–3]. BTI is temperature-activated after a bias voltage has been applied to a Metal Oxide Semiconductor Field Effect Transistor (MOSFET). HCI consists of a high electric field near the drain of a transistor in saturation and brings the change in transistor characteristics. They shift the electrical characteristics of negative-MOS and positive-MOS transistors at operation time. The primary observation of BTI and HCI is an increase in threshold voltage (ΔV_{th}) [4]. Compared with digital ICs, analog ICs are more susceptible to these transistor parameters [5, 6]. Besides, in industry, different from digital ICs [7, 8], the transistor aging effect is directly evaluated by its ΔV_{th} in analog ICs due to their diverse performance specifications.

To achieve **design-for-reliability (DFR)**, the aging analysis is used to predict the occurrence of a circuit failure before errors actually appear in the circuit. The basic principle behind circuit failure prediction is to obtain information about the change of circuit parameters over time and to analyze the obtained data to predict failures. The environmental parameters, process variations, and stresses have an influence on the transistor aging effect [9–11]. The environmental parameters, such as temperature, supply voltage fluctuations, and usage time, have a direct influence on the worst aging degradation. For example, an escalation in temperature correlates with an intensified severity of degradation [10, 11]. The process variations can be modeled by probabilistic models, such as Gaussian process regression and neural processes [12–14]. In industrial practice, the aging degradation attributable to each transistor is quantified in relation to the stresses it experiences, namely the voltage and current applied across its source, drain, gate, and substrate. Consequently, aging analyses are conducted at the level of the transistor netlist, leveraging SPICE simulations to ascertain stress levels by propagating circuit stimuli across each net and branch [15, 16]. In practice, circuit dynamic stimuli (e.g., behavioral signal, frequency divider, and piece-wise linear) and static stimuli (i.e., DC bias) need to be provided for transistor aging analysis [17, 18].

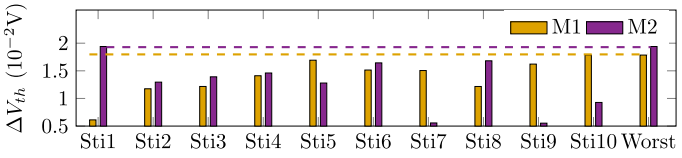
In practical scenarios, the determination of dynamic stimuli, as opposed to static stimuli (e.g., DC bias), proves to be challenging due to their susceptibility to the influence of upstream circuits and the variability of actual input stimuli, among other factors. Consequently, within the industrial context, the worst-case scenario is often employed as a benchmark for assessing reliability and performance [19, 20]. One notable example is graph-based analysis, which is extensively utilized for worst-case timing performance evaluation in static timing analysis [21, 22]. In the assessment of circuit reliability in industry, it is typically necessary to simulate approximately 10 distinct stimuli to ascertain the most severe degradation. Figure 1 shows an example that the worst aging degradation is determined by 10 circuit stimuli to simulate. However, this process incurs substantial costs. First, a significant number of transient steps must be accepted in each SPICE simulation. Second, the requirement to simulate around 10 different stimuli further escalates the resource and time expenditure.

Machine learning (ML) models have the potential to supplant traditional analytical models by facilitating rapid prediction and estimation tasks, once they have been trained on large datasets with accurately labeled ground truth [23, 24]. These models are increasingly being tailored to

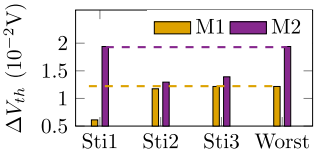


Sti1: VIN In SIN(0 1.1 120MEG 1NS)	Sti2: VIN In SIN(0 0.9 90MEG 0.9NS)
Sti3: VIN In SIN(0 1.2 100MEG 1.1NS)	Sti4: VIN In SIN(0 0.9 110MEG 1NS)
Sti5: VIN In SIN(0 1.2 90MEG 0.9NS)	Sti6: VIN In SIN(0 1 100MEG 1.1NS)
Sti7: VIN In SIN(0 0.9 100MEG 0.9NS)	Sti8: VIN In SIN(0 1 110MEG 1NS)
Sti9: VIN In SIN(0 0.8 120MEG 1NS)	Sti10: VIN In SIN(0 1.1 90MEG 1NS)

(a)



(b)



(c)

Fig. 1. The worst aging degradation and underestimated worst degradations under 10-year usage time. Stimulus definition format example: `SIN(0 1.1 120MEG 1NS)` defines a 120MHz sinusoid with 1.1V amplitude and 1ns delay. M1 and M2 denote two transistors in this circuit. (a) Analog circuit and stimuli. (b) Ten stimuli are given to obtain the worst degradation. (c) Stimuli are reduced to 3 so that the worst degradation of M1 is underestimated (the golden dash line is lower than (b)).

forecast circuit reliability as well [25–28]. For instance, the studies of Jin et al. [26, 27] describe an ML model that utilizes stress factors, such as current density on the metal wire, as input variables to predict electromigration degradation.

The most severe degradation can theoretically be estimated by applying model inference to the stresses induced by each stimulus. However, this process often entails prohibitively long runtimes during the model inference phase, especially since SPICE simulations are resource intensive for large-scale circuits and need to be conducted for 10 different stimuli. In the work of Chen et al. [25, 28], an ML model decouples hard-determined dynamic circuit stimuli from transistor aging reliability analysis to avoid the expensive SPICE simulation. Specifically, an ML model is trained using a dataset generated under a single stress condition with conventional training methods, aiming to predict degradation under varied stress conditions. However, it cannot be directly used to accurately estimate the worst degradations since it is possible that they are not induced by given circuit stimuli. Furthermore, our observations suggest that there is no discernible pattern in the distribution differences between underestimated degradation values and worst-case degradation values. This presents significant obstacles in conducting worst aging analysis because there is no justifiable prior distribution assumption that can be made for the underestimated and worst values. Consequently, modeling the problem with a probabilistic approach, such as using a Gaussian process, becomes challenging.

Intuitively, one might consider training an ML model on a large dataset comprising the most severe degradation cases using traditional training methods. However, gathering data on the worst degradations for this purpose is prohibitively costly. Currently, existing ML techniques are not effectively adapted for the worst aging analysis. To the best of our knowledge, this challenge has not been addressed by any existing works.

A key strategy for efficiently analyzing the worst aging is to reduce the cost of collecting labels. However, the accuracy of labels within the training dataset is compromised, which poses

a significant challenge for training a model to accurately estimate the worst aging degradation. Specifically, the worst degradation cases tend to be underestimated in the training dataset, as they may not be triggered by the circuit stimuli provided for the training. This is illustrated in Figure 1(b) and (c); when only three circuit stimuli are used in the aging simulation, the worst degradation for transistor M1 is not captured accurately.

Domain generalization techniques have been developed to train predictive models on one domain and enable them to generalize effectively to other target domains [29, 30]. Drawing inspiration from the prevalence of similar subcircuits within large-scale analog designs, this article introduces *Wages*—an approach that employs domain generalization to train an ML model on an inaccurate dataset for estimating the worst aging degradation. The contributions of this work are as follows:

— Based on the assumption that transistors exhibit similar worst-case degradation if they share comparable design parameters and are part of similar neighboring subcircuits, we develop a sampling-based method. This method operates within the feature space defined by the transistor and its adjacent subcircuit, and it serves as a substitute for the original labels, representing the worst degradations.
— To ensure a consistent estimation of the worst degradations, the model parameters are updated based on the data from the replaced worst degradation labels while efforts are made to eliminate any pessimistic estimations.
— The model update introduces new feature mappings and label variations. We use these replaced labels to refine the ML model iteratively. Label updates and model updates are performed alternately, enabling the training of the ML model on an inaccurate dataset.
— We conducted experiments on several advanced $5nm$ large-scale industrial designs. The experimental results demonstrate that our method, Wages, can significantly reduce the costs associated with label collection while incurring a negligible estimation error for the worst aging degradations compared to traditional methods.

The rest of the article is organized as follows. Section 2 outlines our problem formulation for estimating the worst aging degradation and provides motivation for the domain generalization methodology, informed by observations from large-scale analog ICs. In Section 3, we introduce Wages, a method for training an ML model on an inaccurate dataset to estimate the worst aging degradation using domain generalization. Section 4 details the data preparation process and describes the ML model used for evaluation. Section 5 discusses the experimental results. Finally, Section 5 concludes the article and suggests directions for future work.

2 Preliminaries

2.1 Problem Formulation

In industry, the worst case can be used as a metric to assess reliability and performance [19, 20, 31]. For evaluating circuit reliability, it is crucial to accurately estimate the worst aging degradation of each transistor prior to finalizing the design for manufacturing. In this work, increase in threshold voltage ΔV_{th} between fresh and after 1- or 10-year periods of usage is adopted as the metric for transistor aging degradation [4]. Traditional methods, however, tend to suffer from either low accuracy or high data collection costs. Therefore, this work aims to provide an accurate estimation of the worst aging degradations at a reduced cost of label collection. The worst aging degradation is formally defined as follows.

Definition 1 (The Worst Aging Degradation). The maximum degradation experienced by each transistor when subjected to circuit stimuli over an equivalent period of usage time.

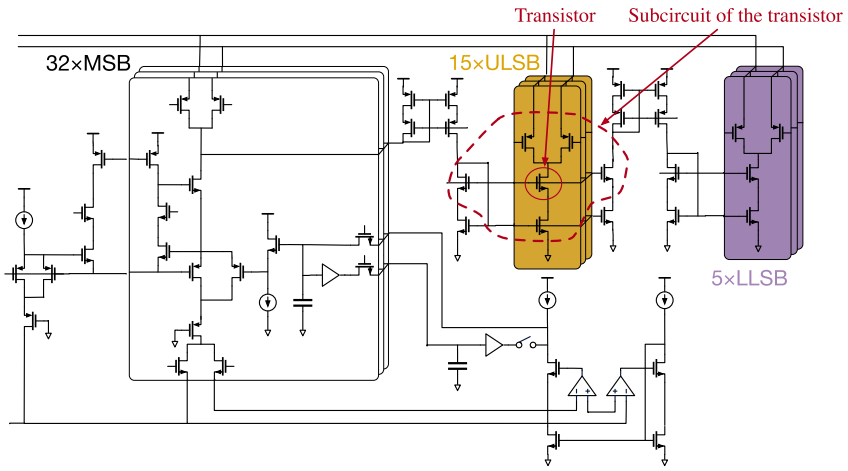


Fig. 2. There are 15 ULSB arrays and 5 LLSB arrays in the large-scale design (14-bit Digital-to-Analog Converter) [32]. They have a similar topology.

In the industry, usage time is often defined according to design specifications and can be set to 1 year or 10 years [8, 25, 28]. There exists a one-to-one mapping between dynamic stimuli, static stimuli, and the circuit's impact on transistor aging degradation. We define the ϵ-hop neighborhood subcircuit of a transistor as follows.

Definition 2 (The ϵ-Hop Neighborhood Subcircuit of a Transistor). A circuit partition that includes the transistor itself and any other devices within a distance of at most ϵ hops from this transistor.

Figure 2 gives one 1-hop neighborhood subcircuit example. Figure 3 gives 1-, 2-, and 3-hop neighborhood subcircuit examples. ϵ is a hyperparameter related to our ML model configurations.

In industry, it is typical for designers to apply 10 specific stimuli to gauge the worst degradation. Our methodology aligns with this industrial practice, using 10 stimuli to determine the worst aging degradation over a consistent usage time. Moreover, to estimate the worst degradations efficiently, it is crucial to decouple them from the detailed aging analysis, thereby circumventing the need for time-consuming SPICE simulations. We will provide a formal definition of our problem formulation.

Problem 1 (The Worst Aging Degradation Estimation). *Given an analog design, static circuit stimuli (i.e., DC bias), and usage time, estimate the worst aging degradation of each transistor.*

2.2 Observation and Motivation

In practice, the worst aging degradations of certain transistors may be underestimated if they are not sufficiently stimulated by the given circuit inputs, as illustrated in Figure 1. However, in large-scale designs, numerous similar subcircuits often exist. For instance, in a 14-bit Digital-to-Analog Converter [32], depicted in Figure 2, there are 15 arrays of the **upper least significant bit (ULSB)** and 5 arrays of the **lower least significant bit (LLSB)**. These arrays share a similar topology but receive different dynamic stimuli from upstream subcircuits, leading to varying aging degradation among the transistors in these subcircuits. Consequently, the potential worst degradations can be identified by examining these differing aging effects.

Domain generalization can be applied to learning from one domain with underestimated worst aging degradations and developing a predictive model capable of estimating the worst

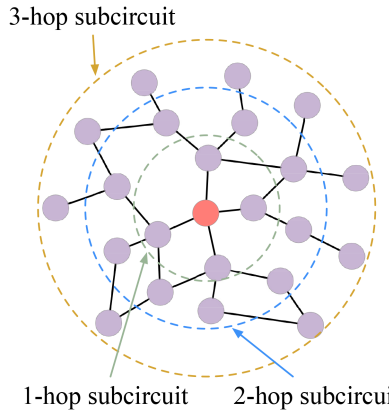


Fig. 3. The neighborhood subcircuit of a transistor.

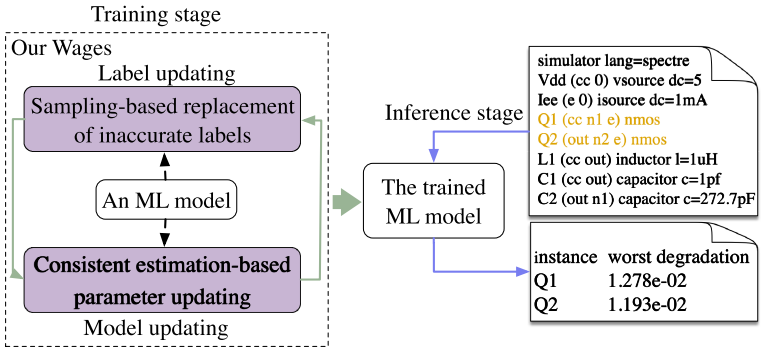


Fig. 4. Our proposed Wages overview.

degradations in another domain [29]. However, two primary challenges arise when employing model generalization to train an ML algorithm on an inaccurate dataset:

— How to replace the (inaccurate) underestimated labels on the training dataset.
— How to use these replaced worst degradations to update model parameters.

3 The Wages Algorithm

3.1 Overall Flow

To minimize label collection costs, we introduce Wages, depicted in Figure 4, which employs domain generalization techniques to train an ML model on a dataset with underestimated labels for worst aging degradation prediction. Our training approach utilizes a sampling-based method in the feature space of the transistor and its ϵ-hop neighborhood subcircuit to replace these underestimated labels. The model parameters are then updated by enforcing consistent estimations of the worst degradation. This process generates new feature mappings and label variations, which are used to iteratively refine the ML model. Through alternating between label updates and model training, the ML model becomes more accurate, even when the dataset initially contains inaccuracies. Ultimately, our proposed method enables the training of an ML model that can accurately estimate the worst aging degradation for each transistor using an initially inaccurate dataset.

3.2 Sampling-Based Replacement of Inaccurate Labels

The performance of an ML model is highly dependent on the accuracy of the training dataset. However, collecting ground-truth data for the worst degradation scenarios is prohibitively expensive. This cost reduction effort results in an inaccurate training dataset. In other words, the worst degradation cases are often underestimated because they may not be triggered by the available circuit stimuli.

In fact, the worst degradations of some unknown transistors are underestimated. $\overline{\boldsymbol{y}} = [\overline{y}_1, \overline{y}_2, \cdots, \overline{y}_N] \in \mathbb{R}^N$ are N transistors' degradations for given circuit stimuli and a same usage time. $\overline{y}_i$ is the worst degradation of the i-th transistor when performing more than one circuit stimuli. We take Figure 1(c) as an example, where three stimuli are given to obtain the worst degradation. As mentioned in Sections 1 and 2, an increase in threshold voltage ΔV_{th} between fresh and after 1- or 10-year periods of usage is used as the aging degradation metric [4]. For example, $\overline{y}_{M1} = 1.23 \times 10^{-2}$ V—that is, the worst degradation among three stimuli, as shown in Figure 1(c). However, according to Figure 1(b), the worst degradation of M1 is underestimated. For convenience, we name these degradations $\overline{\boldsymbol{y}}$ as original labels. Then we need to replace the underestimated worst degradations so that the replaced worst degradation $\hat{\boldsymbol{y}} = [\hat{y}_1, \cdots, \hat{y}_N] \in \mathbb{R}^N$ is not less than the original one—that is, $\hat{y}_i \geq \overline{y}_i$ for $i = 1, \cdots, N$.

In large-scale analog circuits, the presence of many similar subcircuits presents an opportunity to rectify underestimated worst degradation estimates. As shown in Figure 5, the worst degradations for some components can be replaced with values derived from the degradation observed in transistors with similar features and their surrounding subcircuits. Traditional ML models, such as random forests or XGBoost, often require complex feature engineering to extract relevant information from the circuitry. However, these methods lack task-oriented guidance. Recently, neural network models have been introduced to automatically extract features tailored to specific tasks [16, 24]. Therefore, in this work, we employ a neural network model to extract features from transistors and their neighborhood subcircuits.

We begin by representing the feature vector of the i-th transistor and its surrounding subcircuit as $\boldsymbol{f}_i$. This vector can be generated by the initial layers of a neural network model to encapsulate the characteristics of the transistor and its subcircuit. The core concept of this approach is that a neural network model, designed with a task-specific orientation [16, 24], can extract features that accurately represent the transistor and its neighborhood subcircuit within the degradation model. Given that an analog circuit can be naturally depicted as a graph, we utilize a graph neural network to tailor the feature extraction process for the transistor and its subcircuit. It will be introduced in Section 4 in detail. Moreover, the feature vectors of all transistors and their neighborhood subcircuits are collected as a feature matrix $\boldsymbol{F} = [\boldsymbol{f}_1, \cdots, \boldsymbol{f}_N]$.

Based on our observations, similar subcircuits are subject to varying dynamic circuit stimuli, which depend on their upstream counterparts. Consequently, transistors within these similar yet distinct neighborhood subcircuits experience different aging degradations. We propose that the worst degradations can be inferred from the aging degradations observed in these similar subcircuits. Therefore, we hypothesize that a transistor is likely to have a similar extent of worst degradation to another if they share comparable design parameters and are situated in analogous neighborhood subcircuits. In essence, the likelihood of replacing the degradation of one transistor with the more severe degradations of others increases when they exhibit greater similarity in design parameters and neighborhood subcircuits. It is important to note that our approach does not require the adoption of explicit subcircuit identification methods. Instead, we employ a learning-based technique for fuzzy matching of subcircuits, leveraging the potential relationship between the aging effects and the characteristics of the neighborhood subcircuit.

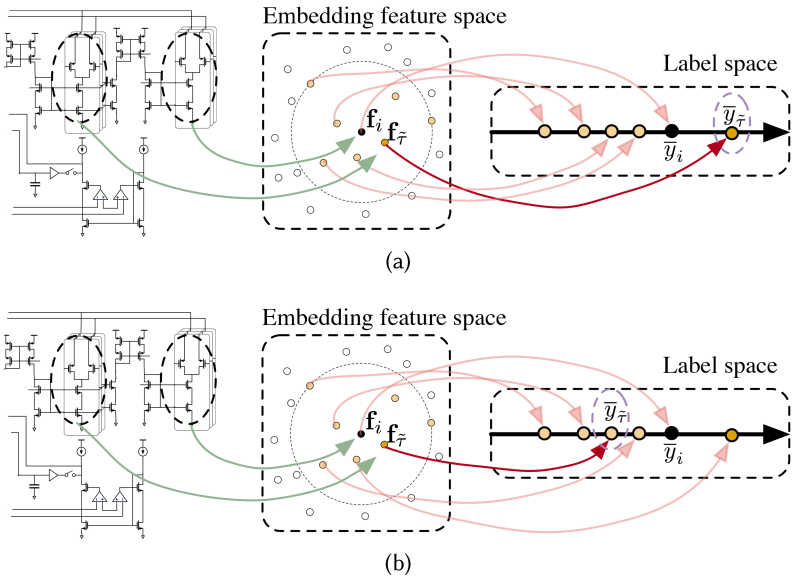


Fig. 5. Worst degradation replacement via sampling: accept scenario (a) and reject scenario (b). The green arrow represents that the transistor and its neighborhood subcircuit are encoded as a feature vector. The red arrow represents the mapping from a feature vector to the worst degradation. The purple and yellow dotted circles denote the transistor and its neighborhood subcircuit.

Based on our assumption, we propose a sampling-based method on the feature space of the transistor and its neighborhood subcircuit to replace the underestimated worst degradations, as shown in Algorithm 1. First, we randomly choose ρ samples from the feature vectors of all transistors and their neighborhood subcircuits (line 1). For each chosen sample $\boldsymbol{f}_i$ within the ρ samples, we cluster its k-nearest neighbors within all feature vectors (line 3). We will randomly sample one neighbor from the k-nearest neighbors and try to replace $\boldsymbol{f}_i$'s degradation. For each neighbor $\boldsymbol{f}_{\tilde{j}}$ in this cluster, the sampling probability is calculated as follows:

$$\mathcal{P}(\boldsymbol{f}_{\tilde{j}})_i = \exp(-d^2_{i,\tilde{j}}/\kappa)/Z, \quad \tilde{j} = 1, \cdots, k, \tag{1}$$

where $d_{i,\tilde{j}} = ||\boldsymbol{f}_i - \boldsymbol{f}_{\tilde{j}}||_2$ is the Euclidean distance between $\boldsymbol{f}_i$ and $\boldsymbol{f}_{\tilde{j}}$. It measures the similarity between two transistors with design parameters and neighborhood subcircuits. κ is a steeping coefficient hyperparameter. Z is a normalization factor.

Based on our assumption, according to the probability, we randomly select one neighbor $\boldsymbol{f}_{\tilde{\tau}}$ (line 4). If its degradation $\overline{y}_{\tilde{\tau}}$ is larger than $\overline{y}_i$, the worst degradation is updated by $\hat{y}_i \leftarrow \overline{y}_{\tilde{\tau}}$. Figure 5(a) illustrates this acceptation scenario. Otherwise, as shown in Figure 5(b), we reject the sample and the worst degradation is the original one—that is, $\hat{y}_i \leftarrow \overline{y}_i$. Algorithm 1 outputs the replaced worst degradations $\hat{\boldsymbol{y}} = [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N]$.

If the inaccurate worst degradations are replaced only once, it brings a low replacement efficiency and inaccurate estimation caused by outliers. Thus, we replace the inaccurate worst degradations M times to generate M worst degradations $\hat{\boldsymbol{y}}^{(m)} = [\hat{y}_1^{(m)}, \cdots, \hat{y}_N^{(m)}]$ such that $\hat{y}_i^{(m)} \geq \overline{y}_i^{(m)}$ and $m = 1, 2, \cdots, M$ denotes the m-th replaced worst degradations.

The ML model must be revised based on these replacements for the worst degradations. However, this model revision introduces variations in feature mappings and labels. To address this, we employ cluster-based sampling to refresh the labels. These updated labels are then used to adjust

ALGORITHM 1: Sampling-based replacement of inaccurate labels.

Input: Subcircuit encoding features $\boldsymbol{F}$ and original degradations $\overline{\boldsymbol{y}}$.

Randomly select ρ samples out of $\boldsymbol{F}$;
for $i = 1$ to ρ **do**
    Cluster the neighborhood subcircuit encoding feature $\boldsymbol{f}_i$ k-nearest neighbors $\boldsymbol{f}_{\tilde{j}}$ and calculate sampling probability by Equation (1);
    According to the probability, sample one $\boldsymbol{f}_{\tilde{\tau}}$ from this cluster and obtain its label $y_{\tilde{\tau}}$;
    if $y_{\tilde{\tau}} > \overline{y}_i$ **then**
        $\hat{y}_i \leftarrow \overline{y}_{\tilde{\tau}}$; ▷ accept
    else
        $\hat{y}_i \leftarrow \overline{y}_i$; ▷ reject
    end if
end for
return The worst degradations $\hat{\boldsymbol{y}}$.

the parameters of the ML model. Nevertheless, the model revision can still result in variations in feature mapping. Consequently, labels must be reevaluated and modified in accordance with the new feature mappings. The processes of label updating and model refinement are carried out alternately and repeatedly to effectively train the ML model despite the initial dataset inaccuracies.

3.3 Consistent Estimation-Based Parameter Updating

Upon acquiring M replaced worst degradations, we will utilize them to revise the parameters of the ML model. Here, we denote an ML model as $f(\boldsymbol{X};\boldsymbol{W})$, where $\boldsymbol{W}$ is its model parameters and $\boldsymbol{X}$ is a feature matrix, representing analog circuit, static circuit stimuli (i.e., DC bias), and usage time. The relationship between $\boldsymbol{X}$ and $\boldsymbol{F}$ is $\boldsymbol{F} = f_L(\boldsymbol{X};\boldsymbol{W})$, where L means the output of first L layers of the ML model $f(\boldsymbol{X};\boldsymbol{W})$. A direct approach involves training the ML model on these replaced worst degradations. However, this strategy may cause the ML model parameters to overfit these particular degradations, leading to overly pessimistic estimations.

To mitigate pessimistic estimations, we propose a parameter updating method based on consistent estimation. The fundamental concept is to update the model parameters using consistent estimations derived from multiple auxiliary models rather than relying solely on the replaced worst degradations. While these models share the same configuration, their parameters are updated using different strategies. This approach can reduce the impact of the substituted worst degradations on the models.

In the proposed method, there are $2 + M$ models: one root model, one generalized model, and M auxiliary models. Their model parameters are denoted as $\boldsymbol{W}^{(r)}$, $\boldsymbol{W}^{(g)}$, and $\boldsymbol{W}_m^{(a)}$ $(m = 1, 2, \cdots, M)$. The generalized and auxiliary models are derived from the root model. Subsequently, the root model is updated to ensure consistent estimations across the generalized and auxiliary models. This approach is designed to minimize the influence of the substituted worst degradations on the generalized model, thereby reducing the likelihood of pessimistic estimations.

As shown in Figure 6, M auxiliary models (the model parameters $\boldsymbol{W}_m^{(a)}; m = 1, 2, \cdots, M$) are derived from the root model (the model parameters $\boldsymbol{W}^{(r)}$) by a gradient method with M replaced worst degradations. The generalized model (the model parameters $\boldsymbol{W}^{(g)}$) is updated by the self-ensembling method [33, 34] and the root model $\boldsymbol{W}^{(r)}$. More specifically, the generalized model parameters $\boldsymbol{W}^{(g)}$ are updated by weight averages of their earlier versions and the root model parameters $\boldsymbol{W}^{(r)}$. Thus, the self-ensembling method can make the generalized model learn from the

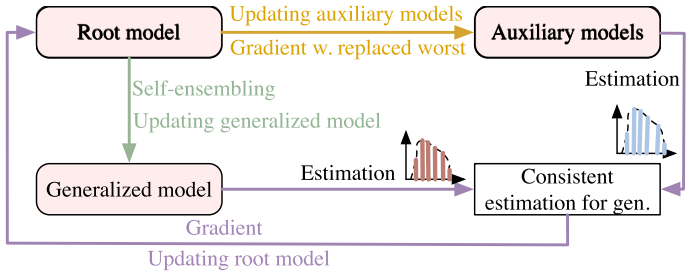


Fig. 6. Consistent estimation-based parameter updating method. Auxiliary models are derived from the root model by a gradient method (yellow arrow). The generalized model is updated by the self-ensembling method and the root model (green arrow). Consistent estimations between auxiliary models and the generalized model are enforced to update the root model (purple arrows).

replaced worst degradations. Consistent estimations between M auxiliary models and the generalized model are enforced to update the root model. According to this strategy, the generalized model with parameters $\boldsymbol{W}^{(g)}$ is not directly updated by the replaced worst degradations so that the pessimistic estimations can be eliminated.

Auxiliary Model Updating. With M replaced worst degradations $\hat{\boldsymbol{y}}^{(m)}$, parameters $\boldsymbol{W}_m^{(a)}$ of M auxiliary model are generated from the root model by the gradient method as follows,

$$\boldsymbol{W}_m^{(a)} = \boldsymbol{W}^{(r)} - \alpha \nabla_{\boldsymbol{W}^{(r)}} \mathcal{L}(f(\boldsymbol{X}; \boldsymbol{W}^{(r)}), \hat{\boldsymbol{y}}^{(m)}), \tag{2}$$

where $\mathcal{L}(f(\boldsymbol{X}; \boldsymbol{W}^{(r)}), \hat{\boldsymbol{y}}^{(m)})$ is the empirical loss function defined as follows,

$$\mathcal{L}(f(\boldsymbol{X}; \boldsymbol{W}^{(r)}), \hat{\boldsymbol{y}}^{(m)}) = \frac{1}{N} ||f(\boldsymbol{X}; \boldsymbol{W}^{(r)}) - \hat{\boldsymbol{y}}^{(m)}||_2^2, \tag{3}$$

which is the **mean square error (MSE)** between the estimated and replaced worst degradations. It is used to enforce the estimated worst degradations $f(\boldsymbol{X}; \boldsymbol{W}^{(r)})$ close to the m-th replaced worst degradations $\hat{\boldsymbol{y}}^{(m)}$. α is the step size.

Generalized Model Updating. Different from auxiliary models, the generalized model's parameters $\boldsymbol{W}^{(g)}$ are updated by the self-ensembling method [33, 34] with the root model's parameters $\boldsymbol{W}^{(r)}$ as follows:

$$\boldsymbol{W}^{(g)} = \gamma \boldsymbol{W}^{(g)} + (1 - \gamma) \boldsymbol{W}^{(r)}, \tag{4}$$

where γ is a smoothing coefficient hyperparameter. According to Equation (4), the generalized model parameters $\boldsymbol{W}^{(g)}$ are updated by weight averages of their earlier versions and the root model parameters $\boldsymbol{W}^{(r)}$. Thus, the self-ensembling method can make the generalized model learn from the replaced worst degradations but eliminate pessimistic estimations.

Root Model Updating. Note that the generalized model's parameters $\boldsymbol{W}^{(g)}$ and M auxiliary model's parameters $\boldsymbol{W}_m^{(a)}$ are derived from root model's parameters $\boldsymbol{W}^{(r)}$. To perform model generalization and update the root model, we first evaluate a consistent estimation between M auxiliary models and the generalized model as follows:

$$\mathcal{J}(\boldsymbol{W}_m^{(a)}) = \frac{\mathcal{KL}(f(\boldsymbol{X}; \boldsymbol{W}^{(g)}), f(\boldsymbol{X}; \boldsymbol{W}_m^{(a)}))}{N}, \tag{5}$$

where $\mathcal{KL}$ denotes the Kullback-Leibler divergence. It measures the difference between two probability distributions of the estimated worst degradations from auxiliary models and the generalized

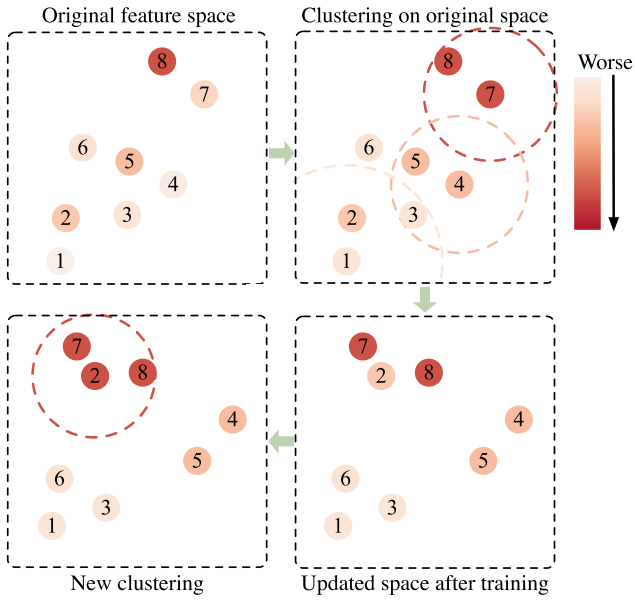


Fig. 7. Feature mapping and label variation after model updating. Sampling in clusters is used to update labels. The new labels are adopted to update the ML model. The updated model brings feature mapping variation. Labels are updated again by sampling according to the features.

model [35]. Furthermore, the average of M consistencies is used as a generalization loss shown as follows:

$$\mathcal{L}_{gen}(\boldsymbol{W}^{(r)}) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{J}(\boldsymbol{W}_m^{(a)}), \tag{6}$$

where

$$\mathcal{J}(\boldsymbol{W}_m^{(a)}) = \mathcal{J}(\boldsymbol{W}^{(r)} - \alpha \nabla_{\boldsymbol{W}^{(r)}} \mathcal{L}(f(\boldsymbol{X}; \boldsymbol{W}^{(r)}), \hat{\boldsymbol{y}}^{(m)}), \tag{7}$$

according to Equation (2). Thus, consistent estimations can be enforced by minimizing the generalization loss $\mathcal{L}_{gen}(\boldsymbol{W}^{(a)})$ to update the root model's parameters with the generalization rate η as follows:

$$\boldsymbol{W}^{(r)} \leftarrow \boldsymbol{W}^{(r)} - \eta \nabla_{\boldsymbol{W}^{(r)}} \mathcal{L}_{gen}(\boldsymbol{W}^{(r)}). \tag{8}$$

3.4 Alternative and Iterative Updating

The ML model is updated using labels that have been revised to reflect the worst degradations. This updated model introduces variations in feature representation, necessitating another round of label replacement as illustrated in Figure 7. Specifically, we employ cluster-based sampling to refresh the labels. These newly updated labels are then used to adjust the ML model's parameters. As a result of this update, the model exhibits variations in its features, which requires us to replace the worst degradations once more in alignment with the modified features. To improve the estimation of the worst aging degradation in a dataset that may be underestimated, we propose a method that alternates and iterates between updating the labels and the model parameters.

First, the root model and the generalized model are initialized by a pretrained ML model. They have the same configurations. The model is trained several epochs on the original inaccurate

dataset. MSE is used as a loss function at the pretraining stage as follows:

$$\mathcal{L}(f(\boldsymbol{X};\boldsymbol{W}^{(p)}),\overline{\boldsymbol{y}}) = \frac{1}{N}||f(\boldsymbol{X};\boldsymbol{W}^{(p)}) - \overline{\boldsymbol{y}}||_2^2, \tag{9}$$

where $\boldsymbol{W}^{(p)}$ denotes the pretrained model parameters. The loss function is used to enforce the estimated worst degradations $f(\boldsymbol{X};\boldsymbol{W}^{(r)})$ close to the original worst degradations $\overline{\boldsymbol{y}}$. It is minimized by a gradient method with the learning rate β to update pretrained model parameters $\boldsymbol{W}^{(p)}$ as follows:

$$\boldsymbol{W}^{(p)} \leftarrow \boldsymbol{W}^{(p)} - \beta\nabla_{\boldsymbol{W}^{(p)}}\mathcal{L}(f(\boldsymbol{X};\boldsymbol{W}^{(p)}),\overline{\boldsymbol{y}}). \tag{10}$$

Sampling-based label replacement and consistent estimation-based parameter updating are then performed alternately and iteratively. Ultimately, the parameters of the generalized model, denoted as $\boldsymbol{W}^{(g)}$, are produced to estimate the worst degradations during the inference stage. This model yields more accurate estimations compared to the root model and reduces pessimistic estimations. This improvement occurs because it is updated using a self-ensembling method, which is less directly influenced by the substituted worst degradations. Our Wages flow is summarized in Algorithm 2.

ALGORITHM 2: Our Wages flow.

Input: The original training set $\overline{\mathcal{D}} = \{\boldsymbol{X}, \overline{\boldsymbol{y}}\}$.

1: Pretrain a model to obtain $\boldsymbol{W}^{(p)}$; ▷ Equation (10)
2: Initialize root model's parameters $\boldsymbol{W}^{(r)} \leftarrow \boldsymbol{W}^{(p)}$, initialize generalized model's parameters $\boldsymbol{W}^{(g)} \leftarrow \boldsymbol{W}^{(p)}$;
3: **for** each generalization epoch **do**
4: **for** $m = 1 : M$ **do**
5: Replace worst degradations $\hat{\boldsymbol{y}}^{(m)}$; ▷ Algorithm 1
6: Compute auxiliary model parameters $\boldsymbol{W}_m^{(a)}$; ▷ Equation (2)
7: Evaluate consistent estimation $\mathcal{J}(\boldsymbol{W}_m^{(a)})$; ▷ Equation (5)
8: **end for**
9: Evaluate $\mathcal{L}_{gen}(\boldsymbol{W}^{(r)})$; ▷ Equation (6)
10: Update root model parameters $\boldsymbol{W}^{(r)}$; ▷ Equation (8)
11: Compute generalized model parameters $\boldsymbol{W}^{(g)}$; ▷ Equation (4)
12: **end for**
13: **return** Generalized model with parameters $\boldsymbol{W}^{(g)}$.

4 Data Preparation and ML Model

4.1 Data Preparation

Netlist Representation. Transistor aging analysis is conducted at the transistor-level netlist [15, 17, 18]. A netlist intrinsically represents a hypergraph [37]. In this representation, each device, **direct current (DC)** source, or **ground (GND)** is considered a node, and each net is viewed as a hyperedge, as depicted in Figure 8(a) and (b). It can be readily demonstrated that constructing a hypergraph has a computational complexity of $\mathcal{O}(p+q)$, where p and q denote the numbers of nodes and hyperedges, respectively. In contrast, the existing directed multigraph representation [25, 28, 36], as shown in Figure 8(c), requires the removal of all nets while preserving the connections, resulting in a computational complexity of $\mathcal{O}(p^2q)$. Consequently, the traditional directed graph representation involves an additional graph transformation step, incurring extra runtime overhead.

When compared with the directed multigraph construction [28, 36], the hypergraph construction is computationally less complex.

Formally, the hypergraph representation transfers a transistor-level netlist to a hypergraph $\mathbb{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ and $\mathcal{E}$ are the node set and hyperedge set. In particular, $\mathcal{R}$ represents the set of connection pin types since different pins need to be distinguished and they play an important role in transistor aging. To distinguish the type of connection, $\Psi = \{(v, e, r)\}$ represents the set of tuples and each element contains a node $v \in \mathcal{V}$, a hyperedge $e \in \mathcal{E}$, and their connection pin type $r \in \mathcal{R}$. In practice, the connection pin types contain gate, drain, source, substrate, anode, cathode, and others [38].

Example 1. Figure 8(a) and (b) provide a concrete example. The node set contains all devices, DC source, and GND—that is, $\mathcal{V} = \{VDD, M1, M2, M3, R1, R2, GND\}$. The hyperedge set is $\mathcal{E} = \{n1, n2, n3, n4, n5\}$. The set of connection pin types contains gate, source, drain, and others—that is, $\mathcal{R} = \{g, s, d, otr\}$. $(M1, n2, g) \in \Psi$ since $M1$ is connected to $n2$ via g (gate).

In addition to circuit topology, parameters of all devices, DC, and GND must be collected. For example, the channel length is one of the transistor's parameters and the resistance value is a resistor's parameter. All parameters of each device, DC source, or GND are concatenated as their feature vector. We denote the feature vector of the i-th device as $\boldsymbol{f}_i^{(0)}$. All feature vectors in an analog circuit are stacked as a feature matrix $\boldsymbol{F}^{(0)}$.

Aging Degradation Generation. To train an ML model, we must generate a label indicating aging degradation for each transistor to form a training dataset. In each design, the aging degradation for every transistor is directly obtained from an industrial aging DFR tool, using a single circuit stimulus provided by expert designers to minimize the cost of label collection. However, this approach may yield inaccurate labels in the training dataset. More specifically, the worst degradation cases, denoted by ΔV_{th}, are likely to be underestimated in the training dataset because they may not be triggered by the particular circuit stimulus applied.

To evaluate an ML model's performance, it is necessary to acquire the ground-truth worst degradations. In industry, achieving a well-balanced tradeoff between simulation time and aging reliability involves applying 10 different circuit stimuli for the same usage time to capture the worst-case degradations for aging reliability assessment. Therefore, based on expert designers' specifications, we adhere to this industrial convention and randomly generate 10 distinct circuit stimuli for each design. Static circuit stimuli are easier to specify and, as such, remain fixed in practice. Dynamic circuit stimuli, however, present a more significant challenge in determining accurately. Consequently, while the static stimuli are kept constant, we randomly assign 10 different dynamic stimuli to each transistor to elicit 10 varied aging degradations ΔV_{th} using the aging DFR tool, as illustrated in Figure 1.

In industrial aging analysis, each dynamic circuit stimulus is defined by its type and parameters. For example, in Figure 1, `SIN(0 1.1 120MEG 1NS)` specifies a 120MHz sinusoid with 1.1V amplitude and 1ns delay. The type of stimulus is hard to deform, whereas the parameters of the stimulus are easy to disturb. Consequently, based on the circuit stimuli provided by expert designers, we introduce random variations to all parameters within the dynamic circuit stimulus, including pulse characteristics like fall time, rise time, and width, as well as frequency divider attributes like ratio, transition time, and average delay. Each parameter undergoes a random shift that conforms to a standard normal distribution and is then scaled by a factor equivalent to the parameter's original value. The worst degradation ΔV_{th}, or ground truth, for each transistor is determined by taking the maximum value from the 10 distinct degradation measurements.

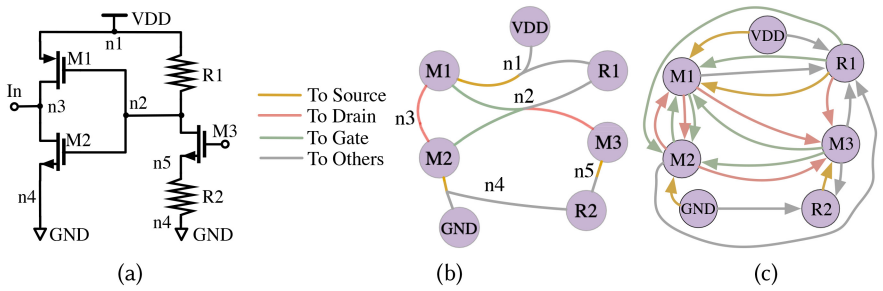


Fig. 8. (a) Netlist. (b) Hypergraph. (c) Directed multigraph [28, 36].

4.2 ML Model

An analog circuit is modeled as a hypergraph. However, this hypergraph constitutes irregular grid-based data, which do not align as neatly with the operations of convolution and pooling found in convolutional neural networks [23, 24]. To address this, graph neural networks have been developed to handle ML tasks on such irregular data structures [39, 40]. Leveraging the hypergraph representation, we have devised a custom node embedding method that encapsulates each device and its surrounding topology (the subcircuit) into a feature vector.

To distinguish among different pins, based on the traditional hypergraph [41], we introduce $|\mathcal{R}|$ incidence matrices—that is, $\boldsymbol{H}_r \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ for $\forall r \in \mathcal{R}$, where $|\cdot|$ is the set cardinality. Each connection pin type has one incidence matrix. The topology is encoded as $|\mathcal{R}|$ incidence matrices $\boldsymbol{H}_r$, with entries defined as follows:

$$h(v,e)_r = \begin{cases} 1, & (v,e,r) \in \Psi \\ 0, & \text{o.w.} \end{cases}, \tag{11}$$

where $v \in \mathcal{V}$ and $e \in \mathcal{E}$. Equation (11) indicates that the entry value is 1 if a hyperedge e is connected to a node v via its pin r.

Example 2. Figure 8(b) and Equation (12) provide a concrete example. According to Figure 8(b), only devices $M1$ and $M2$ are connected to the net $n2$ via g (gate). Therefore, the corresponding entries are 1 and the rest are 0:

$$\boldsymbol{H}_g = \begin{array}{c} \\ \text{M1} \\ \text{M2} \\ \text{M3} \\ \text{R1} \\ \text{R2} \\ \text{VDD} \\ \text{GND} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{n1} & \text{n2} & \text{n3} & \text{n4} & \text{n5} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{12}$$

To distinguish among different pin types, we assign $|\mathcal{R}|$ trainable model parameters w_r for $\forall r \in \mathcal{R}$. To aggregate information from neighbors to the node itself in a hypergraph, we customize a node embedding as follows:

$$\boldsymbol{F}_{\mathcal{N}}^{(l-1)} = \left(\sum_{r\in\mathcal{R}} w_r \boldsymbol{H}_r\right)\left(\sum_{r\in\mathcal{R}} \boldsymbol{H}_r\right)^{\top} \boldsymbol{F}^{(l-1)}, \tag{13}$$

$$\boldsymbol{F}^{(l)} = \sigma\left(\text{CONCAT}\left(\boldsymbol{F}_{\mathcal{N}}^{(l-1)}, \boldsymbol{F}^{(l-1)}\right) \cdot \boldsymbol{W}^{(l)}\right), \tag{14}$$

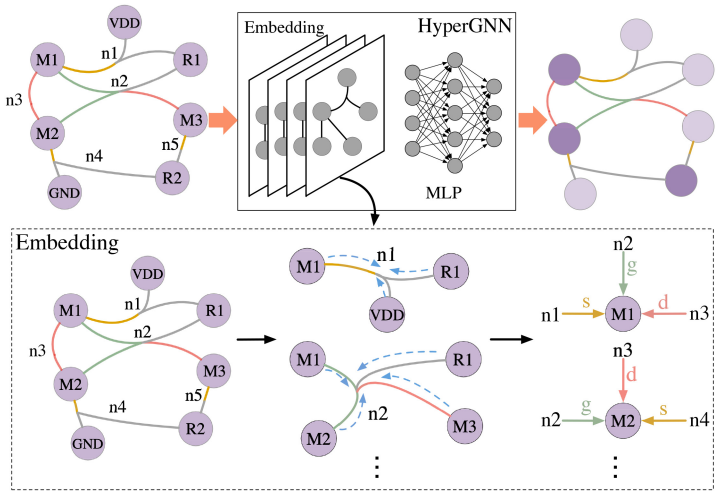


Fig. 9. Node embedding in our HyperGNN.

where $\boldsymbol{F}_{\mathcal{N}}^{(l)}$ is the feature representation of neighboring nodes. $\boldsymbol{F}^{(l)}$ is the feature representation of all nodes. l means the l-th encoding. The initial feature representation $\boldsymbol{F}^{(0)}$ is the device's parameters given from the netlist. For example, the channel length is one of the transistor's parameters. w_r is a trainable model parameter for the type-r pin. CONCAT$(\cdot)$ is the concatenation operation. $\boldsymbol{W}^{(l)}$ is trainable model parameters, and $\sigma(\cdot)$ is ReLU function. Essentially, $\boldsymbol{W}^{(l)}$ and $\sigma(\cdot)$ form a typical **fully connected (FC)** layer to extract features from neighbor nodes and the node itself. As shown in Figure 9, $\sum_{r\in\mathcal{R}} \boldsymbol{H}_r^{\top}$ is used to aggregate information from neighbor nodes to the hyperedge itself. $\sum_{r\in\mathcal{R}} w_r \boldsymbol{H}_r$ is adopted to aggregate information from neighbor hyperedges to the node itself. The node embedding, as shown in Equations (13) and (14), is recursively and sequentially performed L times. Each transistor and its L-hop neighbor topology (subcircuit) are encoded as a feature vector.

Based on our node embedding method, we design an ML model to estimate the worst degradation of each transistor. Our model contains L node embeddings and a **Multilayer Perceptron (MLP)**, where the MLP takes node embedding features and usage time τ as inputs. Since the aging effect is associated with usage time. Our model is simply expressed as $f(\boldsymbol{X};\boldsymbol{W})$, where $\boldsymbol{X} = \{\boldsymbol{H}, \boldsymbol{F}^{(0)}, \tau\}$, $\boldsymbol{H} = \{\boldsymbol{H}_r, \forall r \in \mathcal{R}\}$, and $|\mathcal{R}|$ incidence matrices $\boldsymbol{H}_r$ are used to encode the topology of the hypergraph. $\tau = \{1, 10\}$ since typically, in industry, 1 and 10 aging years are used to evaluate reliability. $\boldsymbol{F}^{(0)}$ is the initial features of all nodes. $\boldsymbol{W}$ denotes model parameters, including w_r and $\boldsymbol{W}^{(l)}$.

In practice, the analysis of large-scale analog circuits introduces scalability challenges due to the substantial amount of training time and memory resources required. The primary cause is the iterative generation of node embeddings from neighboring nodes, as detailed in Equations (13) and (14). Specifically, the embedding features are extracted from an L-hop neighborhood subcircuit. In such cases, the intricate topology of analog circuits results in L-hop neighborhood subcircuits that are extensive in scale. To effectively manage scalability for large-scale graphs, we employ a graph clustering approach [42], supplemented by hypergraph partitioning techniques [37].

The adopted graph clustering method is shown in Algorithm 3. It takes the incidence matrices $\boldsymbol{H} = \sum_{r\in\mathbb{R}} \boldsymbol{H}_r$, feature matrix $\boldsymbol{F}^{(0)}$ of the whole hypergraph representing an analog circuit, usage time τ, and the original worst degradations $\overline{\boldsymbol{y}}$ as inputs. First, the hypergraph partitioning method [37] is utilized to partition all nodes into c non-overlapping clusters $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_c$, where $\mathcal{V}_i$ is the

ALGORITHM 3: The graph clustering method with hypergraph partitioning.

Input: Incidence matrix $\boldsymbol{H} = \sum_{r\in\mathbb{R}} \boldsymbol{H}_r$, feature matrix $\boldsymbol{F}^{(0)}$, usage time τ, and the original worst degradations $\overline{\boldsymbol{y}}$.

1: Construct a hypergraph $\mathbb{G}$ via incidence matrix $\boldsymbol{H}$;
2: Utilize the hypergraph partitioning method [37] to partition all nodes of $\mathbb{G}$ into c non-overlapping clusters $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_c$;
3: **for** Each epoch **do**
4: Randomly select q clusters from c clusters to generate a subgraph $\mathbb{G}_s$ with nodes $\mathcal{V} = \{\mathcal{V}_{t1}, \mathcal{V}_{t2}, \cdots, \mathcal{V}_{tq}\}$ and the worst degradations $\overline{\boldsymbol{y}}_s$ of all nodes in $\mathbb{G}_s$;
5: Extract $\boldsymbol{X}_s = \{\boldsymbol{H}_s, \boldsymbol{F}_s^{(0)}, \tau\}$ from the subgraph $\mathbb{G}_s$ and input $\boldsymbol{X}_s$ and $\overline{\boldsymbol{y}}_s$ input into Algorithm 2 to perform training;
6: **end for**
7: **return** Generalized model with parameters $\boldsymbol{W}^{(g)}$.

i-th cluster node set. Then q clusters are randomly selected to generate a subgraph $\mathbb{G}_s$ with nodes $\mathcal{V} = \{\mathcal{V}_{t1}, \mathcal{V}_{t2}, \cdots, \mathcal{V}_{tq}\}$ and the worst degradations $\overline{\boldsymbol{y}}_s$ of its all transistor nodes for training, where $\mathcal{V}_{ti}$ is the i-th selected node set. The random cluster selection scheme can help prevent information loss that might occur due to hypergraph partitioning [42]. The core strategy hinges on the ability to generate different subgraphs in various training epochs. Consequently, this approach to graph clustering can enhance both robustness and scalability. Moreover, when compared with the neighborhood expansion-based sampling method [28], our method not only reduces runtime but also minimizes information loss. Employing the ML model trained with our proposed Wages on an underestimated dataset, alongside the graph clustering method, we anticipate a substantial reduction in the costs associated with label collection while maintaining a minimal estimation error for the worst aging degradations.

5 Experimental Results

5.1 Benchmarks and Experimental Setting

Given the significant aging-related wear-out observed at the 5*nm* technology node within the industry, our experiments were conducted on eight different industrial **phase-locked loop (PLL)** designs, all implemented using cutting-edge 5*nm* technology. The characteristics of the industrial PLL designs are summarized in Table 1, which indicates that most designs are of a large scale. They are documented in both Spectre and SPICE formats [43]. These designs incorporate 32 distinct types of basic devices and 153 different device parameters. Based on their types and device parameters, we have categorized them into six groups, as depicted in Table 2. This table also enumerates the number of parameters for each device category. Typical parameters include the channel length for transistors and the resistance value for resistors.

To assess the estimation performance and generalization abilities of the model, both inference and training were conducted eight times. In each iteration, one of the eight designs was selected for testing while the remaining designs were used for training. The aging degradation was quantified by measuring the increase in threshold voltage ΔV_{th} between fresh and after 1- or 10-year periods of usage, attributable to HCI and BTI [44].

In our Wages as shown in Algorithm 2, we set the number of nearest neighbors $k = 10$, the predicted degradation number $M = 10$, the steeping coefficient $\kappa = 2$, the step size $\alpha = 0.2$, the smoothing coefficient $\gamma = 0.9$, and the learning rate $\beta = \eta = 10^{-5}$. The generalization epoch is 50. The selected sample number ρ is half the total transistors in the training set. These

Table 1. Description of Our Benchmarks

Design	#Transistors	#Devices	#Nets
1	4,348	99,009	18,155
2	4,382	99,696	18,299
3	3,999	179,758	31,303
4	3,998	185,480	33,819
5	4,980	692,480	111,308
6	523	31,279	6,002
7	6,398	452,109	76,807
8	1,998	96,749	16,006

Table 2. Device Type and Number of Their Parameters

Type	#Parameters
Transistor	51
SPICE Transistor	75
Electrostatic Discharge/Diode	8
Capacitor	12
Resistor	6
DC Source	1

hyperparameters are determined by the grid search method [45]. In other words, the domain of the hyperparameters is divided into a discrete grid. Then, every combination of values of this grid is tried by calculating empirical loss using cross validation. Stochastic gradient descent is used for optimization.

Our ML model is configured by the $L = \epsilon = 3$ node embeddings with 512 output feature size and an MLP, which consists of four FC layers with $4,096$, $4,096$, $1,000$, and 1 output feature size. In the same manner, these configurations are determined by the grid search method [45]. We exploit Algorithm 3 to achieve enough scalability on large-scale graphs. The node number in each cluster is 50, and the random selection number is 20 in Algorithm 3. The number of training epochs is 200.

The feature representation is implemented with Python and NetworkX [48]. All ML models, including our ML model and some baseline models, are implemented with TensorFlow [49], and are trained and tested on a Linux machine with 18 cores and an NVIDIA Tesla V100 GPU with 32GB of memory. We use **mean absolute error (MAE)** and the R2 score [50] to evaluate the absolute and relative accuracy of the testing set. A lower MAE or higher R2 score means better accuracy.

5.2 Comparison with the Traditional Training Method

To demonstrate that our Wages method can provide fast and accurate estimations while reducing the cost of label collection, we trained our ML model using the traditional approach on costly ground-truth degradation data. In contrast, we applied our Wages to aging degradations induced by a single circuit stimulus for each design.

According to Figures 10 and 11, the accuracies of our Wages are very close to the model trained on ground-truth worst degradations by the traditional method. The reasons are that inaccurate labels are calibrated by the proposed sampling-based method on the feature space of the transistor and its neighborhood subcircuit and the model parameters are updated by the proposed consistent estimation for the worst degradation. Label updating and model updating are alternatively performed to train the ML model on the underestimated dataset. In addition, compared with the

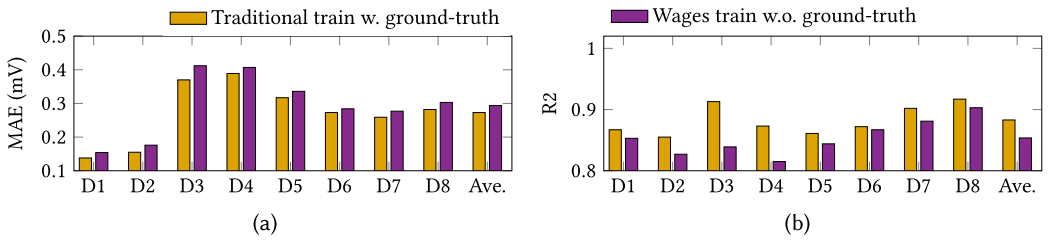


Fig. 10. Accuracies between traditional training with ground-truth and our Wages training w.o. ground-truth in 1-year transistor aging simulation.

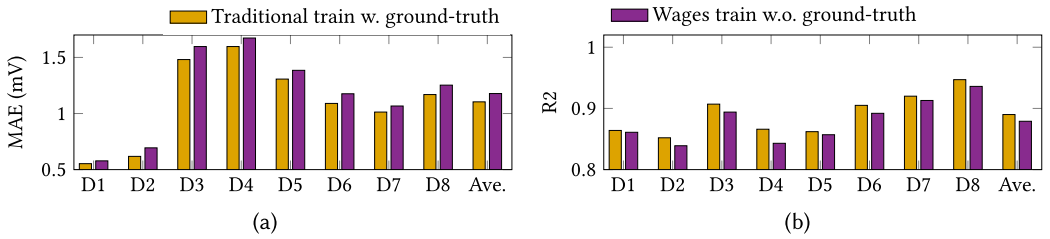


Fig. 11. Accuracies between traditional training with ground-truth and our Wages training w.o. ground-truth in 10-year transistor aging simulation.

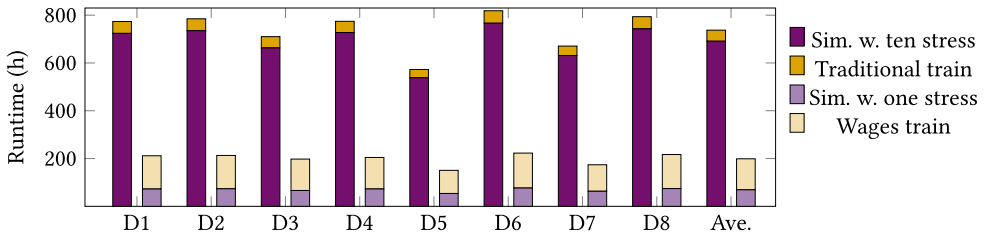


Fig. 12. Running time in training and simulation.

accuracy on the 1-year case as shown in Figure 10, our model can achieve a more accurate relative estimation (R2) on the 10-year case as shown in Figure 11. The main reason is that the 1-year degradation is less than the 10-year degradation, causing more relative estimation error.

We present the running times and their detailed comparison between the traditional method, which relies on ground-truth degradations, and our Wages, which uses aging degradations induced by a single circuit stimulus for each design, as illustrated in Figure 12. The running time for the 1-year worst-case aging estimation model is equivalent to that of the 10-year model; therefore, we only detail the running time for the latter. The phrase "sim. w. ten stress" refers to the running time for 10 simulations needed to acquire ground-truth degradations, and "sim. w. one stress" indicates the running time for a single simulation with only one stimulus. The phrase "typical train" describes the traditional training method where the model is trained on ground-truth degradations and "gen. train" refers to the training method where the model is trained on aging degradations induced by just one stimulus for each design. The left bars represent the total training time for our ML model using the traditional method on ground-truth degradations, which is the sum of "sim. w. ten stress" and "typical train." The right bars represent the total time for our Wages, which is the combination of "sim. w. one stress" and "gen. train."

Table 3. Accuracies (MAE (mV) and R2) among the Industrial DFR Tool and ML Methods in 1 Year

Design				1	2	3	4	5	6	7	8	Ave.
Industrial tool	Static		MAE	1.026	1.041	1.138	1.146	1.080	0.972	1.113	1.136	1.082
			R2	0.169	0.172	0.327	0.318	0.369	0.398	0.291	0.294	0.292
	Dynamic	Random ave.	MAE	0.400	0.467	0.419	0.352	0.394	0.347	0.316	0.324	0.377
			R2	0.661	0.672	0.837	0.823	0.782	0.781	0.843	0.869	0.784
		Expert	MAE	**0.141**	**0.157**	**0.398**	0.411	0.342	**0.224**	**0.248**	0.319	**0.280**
			R2	0.832	**0.849**	**0.863**	**0.848**	0.852	**0.921**	**0.923**	**0.919**	**0.876**
ML methods	GPR [46]		MAE	0.847	0.863	1.196	1.098	1.015	0.987	1.082	1.075	1.020
			R2	0.324	0.459	0.269	0.261	0.309	0.342	0.286	0.303	0.319
	GraphSAGE [39]		MAE	0.419	0.433	0.605	0.727	0.647	0.648	0.509	0.542	0.566
			R2	0.524	0.476	0.458	0.453	0.573	0.568	0.587	0.616	0.532
	RGCN [47]		MAE	0.317	0.364	0.461	0.478	0.536	0.464	0.425	0.453	0.437
			R2	0.732	0.591	0.689	0.726	0.692	0.687	0.722	0.794	0.704
	DHGCN [28]		MAE	0.325	0.366	0.468	0.493	0.534	0.448	0.408	0.428	0.434
			R2	0.698	0.590	0.682	0.729	0.695	0.683	0.784	0.811	0.709
	HyperGCN [41]		MAE	0.402	0.407	0.566	0.635	0.547	0.602	0.440	0.513	0.514
			R2	0.609	0.527	0.603	0.612	0.628	0.619	0.713	0.751	0.633
	Wages		MAE	0.154	0.176	0.412	**0.407**	**0.336**	0.284	0.277	**0.303**	0.294
			R2	**0.853**	0.827	0.839	0.815	0.844	0.867	0.881	0.903	0.854

To train the ML model using ground-truth degradations, 10 simulations with various circuit stimuli are necessary. These simulations significantly increase the running time overhead, making the acquisition of ground-truth degradations exceedingly costly. Conversely, in our Wages method, only a single circuit stimulus is required for each design to conduct a simulation and acquire training labels. As a result, Wages incurs a lower running time overhead for the entire workflow, which includes both simulations and training, even though it introduces a greater running time overhead during the training stage.

5.3 Comparison with the Industrial Aging DFR Tool

We compare the estimation accuracies and running times of the industrial aging DFR tool as indicated in Tables 3 through 5. "Static" and "Dynamic" refer to static and dynamic simulations, respectively [17]. Both types of simulations take a transistor-level netlist and stimuli as inputs. Unlike static simulation, traditional dynamic simulation accounts for dynamic stimuli, including behavioral signals, frequency dividers, and piece-wise linear inputs, thus requiring a significant number of acceptable transient analysis steps. Dynamic simulation is capable of producing a more precise estimation of aging-induced transistor degradation, albeit at the expense of increased computation time. The phrase "random ave." represents the average estimation accuracy obtained from 10 random and varying stimuli using dynamic simulation.

Our Wages method surpasses static aging simulation because the latter does not account for any dynamic circuit stimuli. Furthermore, Wages exceeds the average accuracy of dynamic aging simulations. In addition, Wages can attain accuracy comparable to that of the industrial DFR tool when using circuit stimuli provided by expert designers. When compared to the industrial DFR tool, Wages achieves a significant speedup due to the absence of time-consuming SPICE simulations. As indicated in Table 5, Wages achieves an average speedup of 347.053× over dynamic aging simulations and 60.526× over static aging simulations when using the industrial DFR tool.

5.4 Comparison with Existing ML Models

To compare with the traditional ML method and the state-of-the-art ML models trained by traditional training methods, we implement GraphSAGE [39], DHGCN [28], RGCN [47], and HyperGCN [41] with the same configurations and **Gaussian Process Regression (GPR)** [46], as

Table 4. Accuracies (MAE (mV) and R2) among the Industrial DFR Tool and ML Methods in 10 Years

Design				1	2	3	4	5	6	7	8	Ave.
Industrial tool	Static		MAE	4.133	4.156	4.56	4.582	4.318	3.968	4.48	4.619	4.352
			R2	0.172	0.170	0.315	0.322	0.381	0.417	0.309	0.287	0.297
	Dynamic	Random ave.	MAE	1.578	1.869	1.677	1.487	1.638	1.434	1.292	1.392	1.546
			R2	0.654	0.66	0.825	0.800	0.761	0.774	0.839	0.863	0.772
		Expert	MAE	0.602	**0.667**	1.635	1.683	1.407	**0.935**	**0.994**	1.287	**1.151**
			R2	0.829	**0.855**	0.870	0.837	0.831	**0.917**	**0.920**	0.923	0.873
ML methods	GPR [46]		MAE	3.401	3.447	4.982	4.793	4.460	4.132	4.512	4.603	4.291
			R2	0.312	0.461	0.267	0.252	0.313	0.334	0.281	0.298	0.315
	GraphSAGE [39]		MAE	1.714	1.771	2.819	2.946	2.548	2.633	2.074	2.207	2.339
			R2	0.512	0.483	0.451	0.464	0.580	0.573	0.592	0.625	0.535
	RGCN [47]		MAE	1.312	1.499	1.883	1.951	2.183	1.896	1.742	1.850	1.790
			R2	0.725	0.608	0.701	0.732	0.704	0.692	0.736	0.803	0.713
	DHGCN [28]		MAE	1.339	1.508	1.916	2.004	2.175	1.830	1.677	1.793	1.780
			R2	0.702	0.582	0.685	0.736	0.697	0.688	0.781	0.827	0.712
	HyperGCN [41]		MAE	1.649	1.673	2.318	2.584	2.230	2.447	1.801	2.005	2.088
			R2	0.617	0.531	0.598	0.629	0.635	0.611	0.722	0.764	0.638
	Wages		MAE	**0.578**	0.694	**1.597**	**1.673**	**1.385**	1.176	1.067	**1.253**	1.178
			R2	**0.861**	0.839	**0.894**	**0.843**	**0.857**	0.892	0.913	**0.936**	**0.879**

Table 5. Running Time (s)

Design	Industrial DFR Tool		ML Methods	
	Static	Dynamic	DHGCN [28]	Wages
1	974	23,705	50	47
2	3,759	19,660	50	46
3	6,695	45,526	107	87
4	4,032	22,474	110	85
5	15,347	90,553	472	265
6	499	8,396	30	11
7	11,837	57,074	320	193
8	2,853	16,889	37	29
Ave.	5,750	35,535	147	95
Ratio	60.526	347.053	1.547	1.000

baseline models. Wages trains our model. Besides, the directed multigraph representation [28, 36] is used in GraphSAGE [39], RGCN [47], and DHGCN [28] since they cannot perform ML tasks on a hypergraph. The hypergraph representation is adopted in HyperGCN [41] and our ML model. Different pins are not distinguished in GraphSAGE [39] and HyperGCN [41]. DHGCN [28], RGCN [47], and our ML model can distinguish among them.

Among ML methods, as shown in Tables 3 and 4, our Wages can achieve the best accuracies since underestimated labels bring estimation performance degradation to ML models trained by the traditional method. GPR cannot consider circuit topology information, and this shallow model cannot extract features by task orientation [16, 24]. Besides, the maximum marginal likelihood-based training metric [46] makes it hard to handle inaccurate label scenarios. GraphSAGE [39] and HyperGCN [41] cannot model the analog circuit heterogeneity. However, our Wages adopts a domain generalization methodology to achieve a more accurate estimation of the worst aging degradations. Besides, compared with DHGCN [28], as shown in Table 5, our Wages can achieve 1.547× speedup on average since there is no extra graph transformation in our ML model, whereas DHGCN [28] adopts the directed multigraph representation, where all nets need to be removed while the connections are retained.

6 Conclusion and Future Work

In this article, we introduced *Wages*, a novel approach that utilizes domain generalization techniques to train ML models on inaccurate datasets for the purpose of estimating worst-case aging degradation. This approach is inspired by the prevalence of similar subcircuits within large-scale analog designs. To compensate for unreliable labels, we developed a sampling-based method that operates within the feature space of a transistor and its neighboring subcircuit. Label and model updates were performed alternately, enabling the training of an ML model on the inaccurate dataset. Our approach was validated through several industrial benchmarks on the advanced 5*nm* technology node, demonstrating Wages' capacity for efficient training along with quick and precise estimation of the worst degradation. Our methodology is readily adaptable to other worst reliability analyses.

In the future, we aim to incorporate the ML-based aging reliability model into various design stages, including sizing, placement, and routing, to streamline the design closure process. We also plan to investigate additional techniques to balance runtime and accuracy more effectively. For instance, integrating predictions from the industrial aging DFR tool as supplementary data inputs to the learning model could be beneficial. Customized transfer learning will be explored to adapt a model across different technological contexts. Further consideration must be given to the dynamic and complex nature of circuit degradation prediction, which often involves time-dependent variations in circuit dynamic stimuli, such as behavioral signals and frequency dividers, along with circuit usage time. We are committed to investigating more sophisticated techniques to address this challenging issue.

References

[1] Christian Schlünder, Stefano Aresu, Georg Georgakos, Werner Kanert, Hans Reisinger, Karl Hofmann, and Wolfgang Gustin. 2012. HCI vs. BTI?—Neither one's out. In *Proceedings of the IEEE International Reliability Physics Symposium (IRPS'12)*. IEEE.

[2] Christian Schlünder, Jörg Berthold, Fabian Proebster, Andreas Martin, Wolfgang Gustin, and Hans Reisinger. 2017. On the influence of BTI and HCI on parameter variability. In *Proceedings of the IEEE International Reliability Physics Symposium (IRPS'17)*. IEEE.

[3] Ricardo Reis, Yu Cao, and Gilson Wirth. 2015. *Circuit Design for Reliability*. Springer.

[4] Hussam Amrouch, Behnam Khaleghi, Andreas Gerstlauer, and Jörg Henkel. 2017. Towards aging-induced approximations. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC'17)*. 1–6.

[5] Elie Maricau and Georges Gielen. 2011. Transistor aging-induced degradation of analog circuits: Impact analysis and design guidelines. In *Proceedings of the IEEE European Solid-State Device Research Conference (ESSCIRC'11)*. IEEE, 243–246.

[6] Engin Afacan, Mustafa Berke Yelten, and Günhan Dündar. 2017. Analog design methodologies for reliability in nanoscale CMOS circuits. In *Proceedings of the IEEE International Conference on Synthesis, Modeling, Analysis, and Simulation Methods, and Applications to Circuit Design (SMACD'17)*. IEEE, 1–4.

[7] Yuyang Ye, Tinghuan Chen, Zicheng Wang, Hao Yan, Bei Yu, and Longxing Shi. 2023. Fast and accurate aging-aware cell timing model via graph learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*. Published Online, July 26, 2023.

[8] Yuyang Ye, Tinghuan Chen, Yifei Gao, Hao Yan, Bei Yu, and Longxing Shi. 2023. Aging-aware critical path selection via graph attention networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 12 (2023), 5006–5019. DOI: http://dx.doi.org/10.1109/TCAD.2023.3276944

[9] Elie Maricau and Georges Gielen. 2011. Transistor aging-induced degradation of analog circuits: Impact analysis and design guidelines. In *Proceedings of the IEEE European Solid-State Device Research Conference (ESSCIRC'11)*. 243–246. DOI: http://dx.doi.org/10.1109/ESSCIRC.2011.6044952

[10] Yusuf Leblebici and S.-M. Kang. 1992. Modeling of nMOS transistors for simulation of hot-carrier-induced device and circuit degradation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11, 2 (1992), 235–246.

[11] Joseph B. Bernstein, Moshe Gurfinkel, Xiaojun Li, Jörg Walters, Yoram Shapira, and Michael Talmor. 2006. Electronic circuit reliability modeling. *Microelectronics Reliability* 46, 12 (2006), 1957–1979.

[12] Mengshuo Wang, Wenlong Lv, Fan Yang, Changhao Yan, Wei Cai, Dian Zhou, and Xuan Zeng. 2017. Efficient yield optimization for analog and SRAM circuits via Gaussian process regression and adaptive yield estimation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 10 (2017), 1929–1942.

[13] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. 2018. Conditional neural processes. In *Proceedings of the International Conference on Machine Learning (ICML'18)*. 1704–1713.

[14] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Eslami, and Yee Whye Teh. 2018. Neural processes. *arXiv preprint arXiv:1807.01622* (2018).

[15] François Marc, Benoit Mongellaz, Corinne Bestory, Herve Levi, and Yves Danto. 2006. Improvement of aging simulation of electronic circuits using behavioral modeling. *IEEE Transactions on Device and Materials Reliability* 6, 2 (2006), 228–234.

[16] Tinghuan Chen, Qi Sun, and Bei Yu. 2021. Machine learning in nanometer AMS design-for-reliability. In *Proceedings of the IEEE International Conference on ASIC (ASICON'21)*. IEEE, 1–4.

[17] Ketul B. Sutaria, Pengpeng Ren, Abinash Mohanty, Xixiang Feng, Runsheng Wang, Ru Huang, and Yu Cao. 2015. Duty cycle shift under static/dynamic aging in 28nm HK-MG technology. In *Proceedings of the IEEE International Reliability Physics Symposium (IRPS'15)*. IEEE, CA71–CA75.

[18] Peter M. Lee, Mary M. Kuo, Koichi Seki, P. K. Lo, and C. Hu. 1988. Circuit aging simulator (CAS). In *Proceedings of the IEEE International Electron Devices Meeting (IEDM'88)*. IEEE, 134–137.

[19] Tinghuan Chen, Silu Xiong, Huan He, and Bei Yu. 2023. TRouter: Thermal-driven PCB routing via non-local criss-cross attention networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 10 (2023), 3388–3401.

[20] Siemens. n.d. HyperLynx. Retrieved April 23, 2024 from https://eda.sw.siemens.com/en-US/pcb/hyperlynx/

[21] Yuyang Ye, Tinghuan Chen, Yifei Gao, Hao Yan, Bei Yu, and Longxing Shi. 2023. Graph-learning-driven path-based timing analysis results predictor from graph-based timing analysis. In *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC'23)*. 547–552.

[22] Cadence. 2015. Tempus User Guide. Retrieved April 23, 2024 from https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/tempus-timing-signoff-solution-ds.pdf

[23] Martin Rapp, Hussam Amrouch, Yibo Lin, Bei Yu, David Z. Pan, Marilyn Wolf, and Jörg Henkel. 2022. MLCAD: A survey of research in machine learning for CAD. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 10 (2022), 3162–3181.

[24] Tinghuan Chen, Grace Li Zhang, Bei Yu, Bing Li, and Ulf Schlichtmann. 2023. Machine learning in advanced IC design: A methodological survey. *IEEE Design & Test* 40, 1 (2023), 17–33.

[25] Tinghuan Chen, Qi Sun, Canhui Zhan, Changze Liu, Huatao Yu, and Bei Yu. 2021. Analog IC aging-induced degradation estimation via heterogeneous graph convolutional networks. In *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC'21)*. 898–903.

[26] Wentian Jin, Liang Chen, Sheriff Sadiqbatcha, Shaoyi Peng, and Sheldon X.-D. Tan. 2021. EMGraph: Fast learning-based electromigration analysis for multi-segment interconnect using graph convolution networks. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC'21)*. 919–924.

[27] Wentian Jin, Sheriff Sadiqbatcha, Zeyu Sun, Han Zhou, and Sheldon X.-D. Tan. 2020. EM-GAN: Data-driven fast stress analysis for multi-segment interconnects. In *Proceedings of the IEEE International Conference on Computer Design (ICCD'20)*. IEEE, 296–303.

[28] Tinghuan Chen, Qi Sun, Canhui Zhan, Changze Liu, Huatao Yu, and Bei Yu. 2022. Deep H-GCN: Fast analog IC aging-induced degradation estimation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 7 (2022), 1990–2003.

[29] Fengchun Qiao, Long Zhao, and Xi Peng. 2020. Learning to learn single domain generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'20)*. IEEE, 12556–12565.

[30] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. 2023. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2023), 4396–4415. DOI: http://dx.doi.org/10.1109/TPAMI.2022.3195549

[31] Victor M. van Santen, Hussam Amrouch, and Jörg Henkel. 2019. New worst-case timing for standard cells under aging effects. *IEEE Transactions on Device and Materials Reliability* 19, 1 (2019), 149–158.

[32] Qiuting Huang, Pier Andrea Francese, Chiara Martelli, and Jannik Nielsen. 2004. A 200MS/s 14b 97mW DAC in 0.18μm CMOS. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'04)*. IEEE, 364–532.

[33] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'17)*, Vol. 30.

[34] Geoff French, Michal Mackiewicz, and Mark Fisher. 2018. Self-ensembling for visual domain adaptation. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.
[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
[36] Hao Chen, Keren Zhu, Mingjie Liu, Xiyuan Tang, Nan Sun, and David Z. Pan. 2021. Universal symmetry constraint extraction for analog and mixed-signal circuits with graph neural networks. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC'21)*. 1243–1248.
[37] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1997. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC'97)*. 526–529.
[38] Behzad Razavi. 2002. *Design of Analog CMOS Integrated Circuits*. Tata McGraw-Hill Education.
[39] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'17)*, Vol. 30.
[40] Yuzhe Ma, Zhuolun He, Wei Li, Lu Zhang, and Bei Yu. 2020. Understanding graphs in EDA: From shallow to deep learning. In *Proceedings of the ACM International Symposium on Physical Design (ISPD'20)*. 119–126.
[41] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'19)*. 3558–3565.
[42] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD'19)*. 257–266.
[43] Ken Kundert. 2006. *The Designer's Guide to Spice and Spectre®*. Springer Science & Business Media.
[44] Cadence. 2016. New Generation Reliability Model. Retrieved April 23, 2024 from http://www.mos-ak.org/berkeley_2016/publications/T11_Xie_MOS-AK_Berkeley_2016.pdf
[45] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, 2 (2012), 281–305.
[46] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. Springer.
[47] Junjie Chen, Hongxu Hou, Jing Gao, Yatu Ji, and Tiangang Bai. 2019. RGCN: Recurrent graph convolutional networks for target-dependent sentiment analysis. In *Proceedings of the International Conference on Knowledge Science, Engineering, and Management*. 667–675.
[48] NetworkX. [n.d.]. Network Analysis in Python. Retrieved April 23, 2024 from https://networkx.github.io
[49] TensorFlow. [n.d.]. Home Page. Retrieved April 23, 2024 from https://www.tensorflow.org/
[50] Haoxing Ren, George F. Kokai, Walker J. Turner, and Ting-Sheng Ku. 2020. ParaGraph: Layout parasitics and device parameter prediction using graph neural networks. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC'20)*. 1–6.

Received 18 September 2023; revised 9 March 2024; accepted 6 April 2024