# Aging-Aware Critical Path Selection via Graph Attention Networks

Yuyang Ye, Tinghuan Chen, *Member, IEEE*, Yifei Gao, Hao Yan, *Member, IEEE*, Bei Yu, *Senior Member, IEEE*, and Longxing Shi, *Senior Member, IEEE*

***Abstract*—In advanced technology nodes, aging effects like negative and positive bias temperature instability (NBTI and PBTI) become increasingly significant, making timing closure and optimization more challenging. Unfortunately, conventional critical path (CP) selection tools used in reliability-aware design flow cannot accurately identify CPs under different aging conditions. To address this issue, we propose an aging-aware CP selection flow comprising two parts: 1) critical cell detection and 2) path criticality (PC) computation. We employ graph-attention (GAT) networks to predict the critical cells in the aged circuits, and a PC computation algorithm that takes into account circuit-level and transistor-level parameters to generate PC rank lists. Our experimental results demonstrate that our GAT model outperforms classical machine learning models in detecting critical cells. Additionally, compared with the commercial tool, our aging-aware flow achieves an average accuracy of 99.52%, 98.69%, and 97.20% for top-10%, top-5%, and top-1% path sets, respectively, in five industrial designs subjected to different aging conditions and workloads.**

***Index Terms*—EDA, machine learning, timing analysis.**

## I. Introduction

THE INTEGRATED circuit under advanced technologies causes an increasing demand for design reliability. The most critical reliability bottlenecks for deep-submicron designs are the aging effects of transistors, namely, negative bias temperature instability (NBTI) in PMOS and positive bias temperature instability (PBTI) in NMOS [1], [2], [3], [4], [5], [6]. Path delay can significantly and nonlinearly degrade over time due to aging effects, which have been demonstrated through theoretical analysis and experiments. Eventually, a circuit might exhibit failure if the delay variations on critical paths (CPs) exceed the defined timing constraints.

Manuscript received 10 August 2022; revised 7 January 2023 and 22 March 2023; accepted 24 April 2023. Date of publication 17 May 2023; date of current version 22 November 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62274034 and Grant 61904030, and in part by the Research Grants Council of Hong Kong, SAR, under Project CUHK14209420. This article was recommended by Associate Editor P. Gupta. *(Corresponding authors: Hao Yan; Bei Yu.)*

Yuyang Ye, Yifei Gao, Hao Yan, and Longxing Shi are with the National ASIC Research Center, Southeast University, Nanjing 210096, China (e-mail: yanhao@seu.edu.cn).

Tinghuan Chen was with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR. He is now with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China.

Bei Yu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR (e-mail: byu@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TCAD.2023.3276944

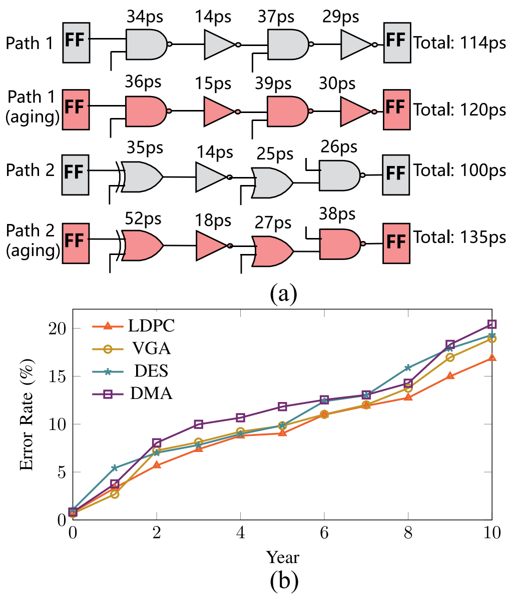


Fig. 1. (a) Example showing how the path, which was uncritical before aging, became critical after aging. (b) Error rate of the top-10% path sets on four benchmarks in 10-year aging experiments.

To solve this issue, a reliability-aware circuit design flow has been developed to predict the design margin accurately [1]. This flow utilizes aging-aware SPICE, which estimates design performance after aging through testbench simulations. SPICE can interface with various commercial application programming interfaces (APIs), including MOS reliability analysis (MOSRA) [7], open modeling interface (OMI) [8], and TSMC modeling interface (TMI) [9], to perform aging-aware timing analysis [5]. However, it is not practical to perform aging-aware SPICE for large-scale circuit netlists while considering the prohibitively expensive computation cost [3]. To balance reliability accuracy and simulation efficiency, aging-aware SPICE must simulate a small set of true CPs.

CP selection is usually performed with ranking slack values in the reliability design flow. For different requirements, the top-10%, top-5%, and even top-1% CPs can be chosen for design optimization. However, typical timing analysis tools (such as PrimeTime [10]) cannot take the aging effects into account. Since different cells exhibit various timing behaviors caused by aging, the CP ranking may be changed [3]. Fig. 1(a) presents an example of two real timing paths (i.e., the delay values are measured using HSPICE), where their timing criticality ranking is changed due to aging effects. It illustrates that the overall path delays have been affected differently while suffering from aging effects in the same circuit. The error rates increase with the timing and reach 16.88%, 18.93%, 19.32%, and 20.43% on the four open-source circuits in 10-year aging experiments, as shown in Fig. 1(b).

A general solution to identify all CPs accurately is aging-aware static timing analysis (STA), which contains two essential parts: 1) workload analysis and 2) aging-aware timing library [3], [11], [12], [13], [14], [15], [16]. Workload analysis estimates signal probability (SP) profiles using a logic simulator with varying testbenches under different working scenarios. And aging-aware timing library creates cell timing-degradation libraries under different aging stress conditions through circuit simulations. By analyzing the circuit timing, it generates CP sets based on cell-level timing results.

However, the accuracy of CP selection results cannot meet the design requirement simply based on the results of aging-aware STA [17]. Aging effects have a strong dependence on different operation conditions, such as supply voltage, temperature, and SP [5]. Usually, these parameters are not spatially or temporally uniform, but vary significantly from cell to cell and from time to time [2]. Due to the complex dependence on operating conditions, the exhaustive aging-aware library with various operating conditions will be extremely expensive in computation and memory, which is impractical in the application [17]. It is still a tremendous challenge to predict the aging-induced timing degradation efficiently and accurately using the aging-aware timing library in aging-aware STA. Thus, the CP selection results are not accurate sufficiently based on cell-level timing results generated via aging-aware STA.

To improve the accuracy of CP selection after aging-aware STA, we shift the focus toward the *critical cell detection* and the *computation of path criticality (PC)*. Critical cells found on real CPs after aging are detected, while PC reflects the impact of the identified critical cells on path slack. By detecting these critical cells and computing PC, circuit PC ranking lists can be accurately and effectively generated for selecting aging-aware CPs. Accurately estimating aging-induced delay degradation with an appropriate size model under complex operating conditions is impractical [17]. However, detecting critical cells based on their relative importance, which is influenced by aging effects and circuit structure, is a feasible alternative [17], [18], [19].

This article proposes an advanced aging-aware CP selection flow. It includes a deep joint representation graph attention network-based (GAT-based) learning framework for detecting critical cells, as well as a practical algorithm for computing PC to rank the CPs. In our learning framework, given a circuit represented as a graph, deep autoencoders automatically generate low-dimensional representations of both the circuit's structure and attributes for each cell using multiple graph-attention (GAT) layers. The high-quality embeddings can capture both circuit's structural and cell attributes information (including aging-related information) jointly. After this, the corresponding decoder functions can reconstruct both the topological structure and nodal attributes. The disparity between the original and the estimated node data (i.e., reconstruction errors) can help find critical cells in the circuits, which can be formulated as an anomaly detection problem on an imbalanced dataset. By using the disparity value, our framework can achieve critical cell detection fast and accurately. Finally, a PC computation algorithm considering critical cell impacts is developed to generate CPs. The key contributions of this article are summarized as follows.

1) We develop an end-to-end dual GAT autoencoder that seamlessly models the attributed circuit networks and conducts critical cell detection in a uniform framework.
2) We propose an algorithm to calculate the PC based on the results of critical cell detection. The higher the criticality of the path, the more necessary it is to perform aging-aware SPICE, and the more important it is to improve the timing performance of the path under aging effects.
3) We leverage a parallel training and inference scheme with multiple GPUs to achieve speedup on the overall modeling process. And the PC computation algorithm is parallelly performed on multithreading.
4) The experimental results indicate the proposed GAT autoencoder outperforms conventional machine learning models in terms of detection accuracy. Besides, the proposed flow can output true CP sets accurately and effectively under different aging conditions. It is helpful to achieve superior design reliability results on industrial designs.

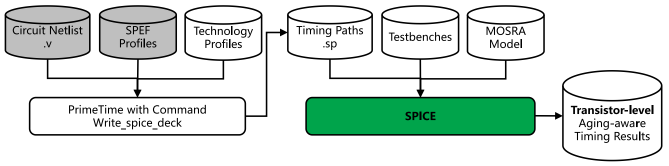


Fig. 2. Aging-aware SPICE simulation flow.

## II. Preliminaries

### A. Aging-Aware SPICE Simulation

Aging-aware SPICE simulation can provide accurate aging-aware timing analysis [5]. As shown in Fig. 2, there are two major steps.

1) Given the circuit netlist and standard parasitic exchange format profiles (SPEF profiles), PrimeTime [10] helps generate SPICE deck templates from the design database for particular paths of interest. The SPICE deck templates are transistor-level netlists of timing paths and includes the capacitive cross-coupling structure and the parasitic information.
2) Based on the SPICE deck templates and different testbenches, the commercial SPICE simulation tool (HSPICE in this article [20]) with the MOSRA aging model [7] is adopted to perform the aging simulation on each extracted timing path.

Then SPICE can generate the transistor-level aging-aware timing results. MOSRA in HSPICE offers an accurate solution

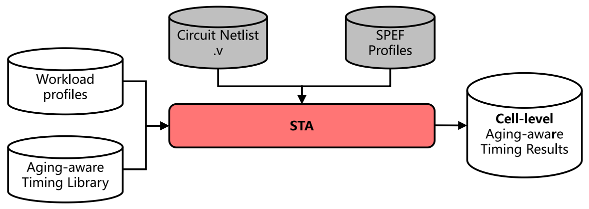


Fig. 3. Aging-aware STA flow. The workload profiles are generated via logic simulator (Modelsim in this work) and the aging-aware timing library is generated via cell-level characterization via PrimeLib.

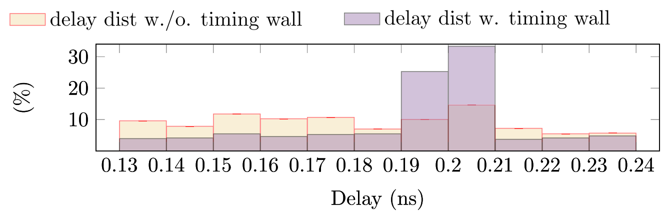


Fig. 4. Distribution of the top-10% CP delays for the design with timing wall (under one global timing constraint) and without timing wall (under multiple timing constraints).

for analyzing the timing degradation. The aging-aware SPICE is accurate but has a prohibitive runtime cost for large-scale circuits.

## B. Aging-Aware STA

Conventional CP selection after aging is directly based on the aging-aware STA [3], [21], [22], [23], as shown in Fig. 3. The aging-aware STA can get all the slacks of timing paths after aging at cell-level in the design, and then CP selection is performed based on the results. Compared with traditional STA flow, aging-aware flow contains two extra parts: 1) workload analysis and 2) cell-level characterization for the aging-aware timing library. There are many studies about workload analysis to obtain signal probabilities and activity factors for all the cells in the circuits under different working testbenches. With commercial EDA tools like Modelsim, their inferred signal probabilities and activity factors have a high accuracy [3], [21], [24]. However, there are only a few studies about cell-level characterization for aging-induced cell delay degradation, which include look-up table (LUT)-based [3], [11], [12], [13], and machine learning-based methods [14], [15], [16].

The LUT based methods use cell-level simulation models to avoid unacceptable runtime but have lower accuracy. In [11], the researchers propose LUT-based gate delay models to capture the impact of NBTI-induced threshold voltage shift of PMOS transistors on the corresponding cell delay degradation. However, both the PBTI effect in NMOS transistors and the slope of rising/falling signals are not considered. Kiamehr et al. [12] proposed to use an aging-aware standard cell library, extending the standard cell library by considering different input signal probabilities. In [3], an accurate LUT-based method is introduced to estimate both NBTI-induced and PBTI-induced delay degradations of each cell. However, all these methods are limited to modeling the operating conditions under which LUT is built. In addition, the model size grows exponentially with the inputs and internal nodes increase in the logic cell.

Machine learning-based methods [14], [15], [16] can make a better tradeoff between the aging-aware timing library accuracy and model size, which are more popular and practical in industrial. Additionally, Synopsys just recently announced PrimeLib [25], a new characterization solution that will also support ML-based cell library characterization. However, the aging effect in each gate in the circuit is independently modeled in classical machine learning methods. Their accuracy cannot always meet reliability-aware design requirements.

## C. Motivation

For a design with a single global timing constraint, the purple part of Fig. 4 shows the distribution of the top-10% CP delays. There is a timing wall problem where most of the top paths have similar slacks. Conversely, for a design with multiple timing constraints, the yellow part of Fig. 4 shows the distribution of the top-10% CP delays. The delay from 0.13 to 0.24 ns exhibits a uniform distribution and the timing wall problem is smoothed. According to [24], [26], and [27], designs without timing wall problems are also very common and reasonable in the reliability-aware design flow. In the classical design flow, the synthesis tool with a global timing constraint only focuses on the CP and allows other paths to become near-critical to optimize area and power. In the reliability-aware design flow, the synthesis tool with multiple timing constraints allows naturally fast paths to obtain smaller delay values and larger timing margins. Hence, the timing constraint greatly varies depending on the set of paths being considered. It is crucial to accurately identify the aged top-$K$% paths and simulate them via aging-aware SPICE to acquire accurate delay values. Performing timing optimization manually on these paths becomes necessary at the end stage of the design flow. Aging-aware STA offers less accuracy as compared to aging-aware SPICE in large-scale designs [10]. Thus, CP selection through aging-aware STA may lead to inaccurate results. Balancing accuracy with runtime cost poses a challenging tradeoff.

We summarize two problems of conventional aging-aware STA while applying them to CP selection.

1) All the LUT-based and conventional machine learning-based aging-aware timing libraries model the delay degradation of aging for each cell independently. These methods give limited considerations about circuit structure. However, the delay degradation of the target gate in circuits is influenced by all neighbor cells significantly while considering aging effects [2], [5]. Therefore, these methods are not sufficient to predict the aging-induced delay degradation accurately when the circuit structure is complex in advanced technologies.
2) In the progress of aging-aware STA, the timing analysis is achieved on the cell-level. Thus, transistor-level process parameters in the technology profiles are totally ignored. However, the aging-induced delay is sensitive to process parameters under some extreme aging conditions (long runtime and large workloads) [5]. It makes the limited accuracy of cell-level timing results generated via aging-aware STA.

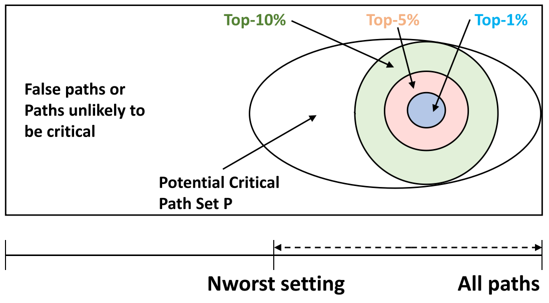

Top-10%
Top-5%
Top-1%
False paths or Paths unlikely to be critical
Potential Critical Path Set P
Nworst setting
All paths

Fig. 5. Relationships between the used potential CP set $\mathcal{P}$ and all paths.

### D. Potential Critical Path Set and Top-K% Path Set

The number of potential CPs that reside in a slack threshold can be prohibitively large. It causes an extremely low efficiency while achieving true CP selection. In this work, we focus on ensuring timing margins and adjusting timing constraints after aging through aging-aware CP selection. Thus, a large number of paths that are unlikely to cause a delay problem (like timing violation) and false paths can be removed. The potential CP set $\mathcal{P}$ that we generate via PrimeTime, consists of the worst 10 paths at each endpoint (nworst 10) and the worst path through each cell (nworst 1). Identical paths are merged, and top-$K$% paths are paths with top-$K$% smallest slacks under aging-aware SPICE simulation within the potential CP set. The relationships among the top-$K$% path set, the potential CP set $\mathcal{P}$ and all paths are shown in Fig. 5. Moreover, the size of $\mathcal{P}$ is determined by the experimental settings nworst on each endpoint and cell, and influences the efficiency and accuracy of CP selection. To ensure appropriate settings, we employ results from transition fault testing to generate potential CPs, similar to the approach proposed in [28].

### E. Problem Formulation

We focus on figuring out the aging-affected CPs fast and accurately, to improve the design reliability and efficiency. The problem formulation is shown as follows.

*Problem 1 (Aging-Aware CP Selection):* Given a design of netlist, technology profiles, power consumption profiles, SPEF profiles, workload files, and timing reports via aging-aware STA, accurately select the CPs with small runtime costs after aging happens under specific conditions.

### F. Overall Flow

In this article, Problem 1 is divided into two tasks: 1) critical cell detection, which are cells on the CPs and 2) PC computation, which is based on critical cells and other cross-layer parameters in the circuit. From the perspective of recent machine learning research, critical cell detection should be cast as an anomaly detection task. Since the circuit netlist is able to be represented as a graph easily, we use GAT to learn the latent representations of gates. In order to adopt a graph learning method, we model a circuit as an attributed network. An attributed network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$ is defined as an undirected graph consisting of: 1) a node set $\mathcal{V} = \{v^{(1)}, v^{(2)}, \ldots, v^{(n)}\}$, where $|\mathcal{V}| = n$; 2) an edge set $\mathcal{E}$, where $|\mathcal{E}| = m$; and 3) node attributes $\boldsymbol{X} \in \mathbb{R}^{n\times k}$, where $i$th row vector $\boldsymbol{x}_i \in \mathbb{R}^k (i = 1, \ldots, k)$ is the attribute information for the $i$th node. In this article, the attributed network $\mathcal{G}$ is represented by an attribute matrix $\boldsymbol{X}$ for node features, an adjacency matrix $\boldsymbol{A}$ for topological structure, and a label vector $\boldsymbol{y}$. In addition, we give the definition of the CP and cell here.

*Definition 1 (CP):* The timing paths with top-$K$% smallest path slacks within the potential CP set under aging-aware SPICE simulation.

*Definition 2 (Critical Cell):* The cells are on CPs.

*Task 1 (Critical Cell Detection):* Given a set of circuit netlists represented by attributed networks, our objective is to detect the anomalous nodes (critical cells), which differ significantly from the majority of the reference normal nodes in circuits (uncritical cells) while considering the structure and attribute information.

*Task 2 (PC Computation:)* Given a potential CP set generated via aging-aware STA, our objective is to quantify the PC of each timing path in the set accurately based on the critical gate detection results and other transistor-level parameters. CPs can be selected using the final PC ranking lists.

## III. GAT-Based Critical Cell Detection

In this section, we introduce the critical cell detection framework in detail. Each cell on top-$K$% paths in the potential CP set is regarded as an anomaly. The value $K$ is flexible. As shown in Fig. 6, the framework consists of three essential components: 1) deep structure autoencoder, nonlinear attribute autoencoder and 2) logistic classifier. The deep structure autoencoder contains two parts, including a structure encoder and a decoder. The encoder comprehensively captures the latent structure information of circuits and generates new node embeddings $\boldsymbol{E}$ by taking the original circuit attribute networks $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$ as input. The decoder is used to reconstruct the original circuit structure based on the embedding results $\boldsymbol{E}$ and outputs the reconstruction error $\boldsymbol{R}_A$ from a circuit network structure perspective. Similar to deep structure autoencoder, nonlinear attribute autoencoder generates attribute embeddings $\boldsymbol{Z}$ for cell attributes $\boldsymbol{X}$ through attribute encoder and reconstructs original attribute information through attribute decoder. It can output the reconstruction error $\boldsymbol{R}_X$ from the cell attributes perspective. Taking reconstruction error from both circuit structure and cell attributes $\boldsymbol{S}$ as input, a logistic classifier is used to find critical cells whose joint reconstruction errors are much larger than others.

### A. Initial Dataset Generation

*Original Attributes:* Before leveraging GAT in the graph learning framework, the original attribute matrix $\boldsymbol{X}$ needs to be given. These initial attributes are related to the timing aging effect, including transistor-level and cell-level information, which have an important influence on aging-aware critical cell detection. The transistor aging phenomenon occurs due to the formation of interface traps (breaking of $S_i - H$ bonds at the $S_i - S_iO_2$ interface) and oxide traps (capturing of charges in the oxide vacancies within the dielectric). For all cells in the netlist, the aging effects under various operating

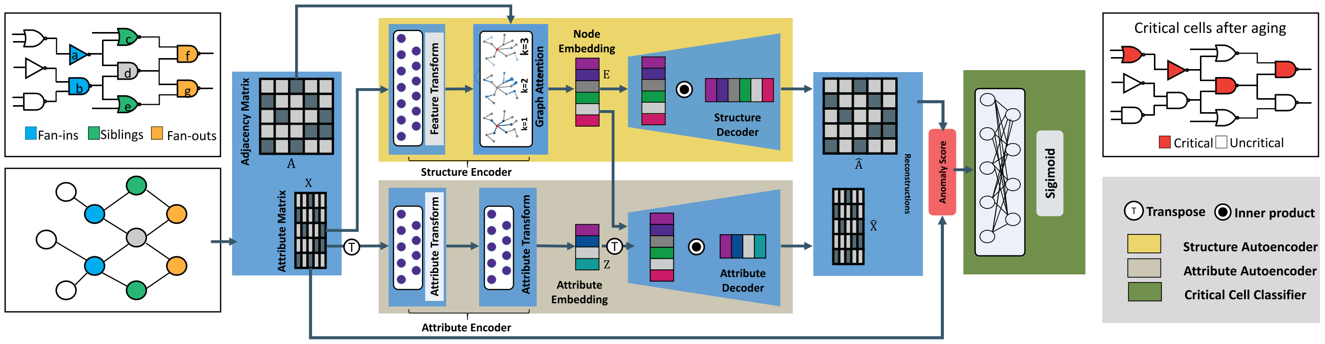


Fig. 6. Framework of GAT-based critical cell detection. The input is a netlist graph represented in an adjacency matrix and its original node attributes. There are three key parts: circuit structure deep autoencoder, circuit attribute deep autoencoder, and critical cell detection classifier. Note that the GAT layers help us to get node embedding in the circuit structure deep autoencoder.

TABLE I
ORIGINAL CELL ATTRIBUTES USED IN OUR GAT

| location | type | description |
|---|---|---|
| targets | working temp. | working temperature |
| | threshold voltage | threshold voltage assignments |
| | total signal pro. | sum of signal probability of input pins |
| | wst signal pro. | worst signal probability of input pins |
| | wst output slack | worst slack of output pin |
| | wst input slack | worst slack of input pins |
| | max output slew | max transition of output pin |
| | max input slew | max transition of input pins |
| | tot input cap | sum of input pin cap |
| | tot output cap | output net cap + tot fanout cap |
| fanins | tot fanin cap | sum cap of fanins |
| | wst fanin slack | worst slack of fanins |
| | max fi. output slew | max trans. of fanin output pins |
| | avg fi. output slew | average trans. of fanin output pin |
| fanouts | tot fanout slack | sum of slack of fanouts |
| | wst fanout cap | worst input pin cap of fanouts |
| | tot fanout cap | sum of input pin cap of fanouts |
| | avg fanout cap | average input pin cap of fanouts |
| siblings | tot sibling cap | sum of input pin cap of siblings |
| | wst sibling cap | worst input pin cap of siblings |
| | tot sibling slack | sum of slack of siblings |
| | wst sibling slack | worst slack of siblings |

conditions cause different delay degradations due to trap generation. Thus, we select some important circuit parameters to represent initial attributes based on domain knowledge and parameter sweeping experiments, as shown in Table I. Typically, the attributes contain circuit, operational, and timing information. This information is from the netlist, power consumption profiles, workload files, SPEF files, and timing results via aging-aware STA.

The cell slack plays an important role when CP selection in classical verification flow, which is the main reason for timing failures.

The signal probability has a great impact on trap generation. Higher SP of cells causes a greater operation cycle of transistors. As a result, more interface traps and oxide traps will be generated inside the transistors of the cell, i.e., the aging rate of the cell will increase. The SP is workload dependent [29]. We follow the previous method [24], [30] to obtain the SP of each cell from workload profiles. In order to generate workload profiles, we run logic simulations using Modelsim [31] based on some input vectors. As there is no specific application for our benchmark circuits and the workload is always highly unpredictable during the design stage, the input vectors used in our work are generated randomly. For evaluating the efficiency of our proposed technique under different workloads, we test our work when the average signal probabilities are different. Note that the workload in our work can be changed for other representative applications, which can be obtained by system-level definitions.

The working temperature impacts the speed of transistor trap generation significantly. As it increases, the rate of interface and oxide trap generation increases, bringing threshold voltage to increase faster and the transistor's mobility to decrease faster. This includes both the PBTI effect in NMOS transistors and the NBTI effect in PMOS transistors. The temperature and power consumption of a chip are tightly coupled. Thus, we follow the previous method [21], [24], [29] to get the temperature of each cell based on power consumption profiles extracted from Design Compiler [32]. The detailed progress is introduced as follows: 1) divide the chip layout into several rectangular grids; 2) estimate the leakage and dynamic power of each cell in the circuit based on workload profiles and SPEF profiles using Design Compiler; 3) generate the power consumption files for each cell by adding the leakage and dynamic power; and 4) obtain temperatures of each cell based on power consumption profiles using the publicly available tool HotSpot [33].

The input signal slew and outputload capacitance have an influence on threshold voltage degradation and transistor mobility. Thus, they should be considered while analyzing aging-induced delay degradation.

There are more details of the attributes illustrated in Fig. 6. For determining the initial attributes of a target cell $d$, we take the information of its fanins {a, b}, siblings {c, e}, and fanouts {f, g} into account. However, these manually engineered features are not sufficient to detect critical cells. To get better node representations, we leverage GAT to perform graph representation learning.

*Labels:* Every node has a binary label $y^{(v)}$, $v \in \mathcal{V}$. We set "0" (negative) as uncritical and "1" (positive) as critical. Critical cells are located on the top-$K$% paths, while the remaining cells are deemed uncritical. The value of $K$ can be adjusted to meet different requirements. Labels can be obtained from aging-aware SPICE flow (as shown in Fig. 2).

**Algorithm 1** Circuit Node Structure Embedding Methodology

```
Require: Circuit netlist graph: $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$; Adjacency matrix: $\boldsymbol{A}$; Node attributes matrix: $\boldsymbol{X} : \{\boldsymbol{x}^{(v)} : \forall v \in \mathcal{V}\}$; Search depth: $D$; Weight matrices and bias of transform layer: $\boldsymbol{W}_T$ and $\boldsymbol{b}_T$; Weight matrices of GAT-layers: $\{\boldsymbol{W}_d, \forall d \in \{1, ..., D\}\}$.
Ensure: Node structure embedding: $\boldsymbol{E} : \{\boldsymbol{e}_D^{(v)} : \forall v \in \mathcal{V}\}$.
1: $\boldsymbol{e}_0^{(v)} \leftarrow ReLU(\boldsymbol{W}_T \boldsymbol{x}^{(v)} + \boldsymbol{b}_T), \forall v \in \mathcal{V}$; ▷ Transformer layer
2: for $d \leftarrow 1$ to $D$ do
3:     for $v \in \mathcal{V}$ do
4:         Compute $\boldsymbol{\alpha}_{d-1}^{(vu)}$ via Equation (1), $u \in \mathcal{N}_v$;
5:         $\boldsymbol{g}_d^{(v)} \leftarrow \boldsymbol{e}_{d-1}^{(v)} + \sum_{u \in \mathcal{N}_v} \alpha_{d-1}^{(vu)} \cdot \boldsymbol{e}_{d-1}^{(u)}$ ▷ Aggregation
6:         $\boldsymbol{e}_d^{(v)} \leftarrow \sigma\left(\boldsymbol{W}_d \cdot g_d^{(v)}\right)$; ▷ Encoding
7:     end for
8: end for
```

## B. Circuit Structure Deep Autoencoder

As shown in Fig. 6, the circuit structure deep autoencoder contains two important parts: 1) *structure encoder* and 2) *structure decoder*. For the encoder part, it encodes circuit graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$, represented with a adjacency matrix $\boldsymbol{A}$ and cell attribute matrix $\boldsymbol{X}$, into the latent node embedding $\boldsymbol{E}$. It consists of an attribute transform layer and GAT layers. For the decoder component, it tries to reconstruct the original circuit structure based on the latent node embedding $\boldsymbol{E}$ generated through the encoding network. After encoding and decoding, we can get the reconstruction error $\boldsymbol{R}_A$ for each cell. Especially, in GAT layers, multiple layers of encoding functions are stacked to extract features from multihop neighborhoods.

*Structure Encoder:* In the first step of structure encoder, we use one learnable layer to transform the initial cell attributes $\boldsymbol{X}$:$\{\boldsymbol{x}^{(v)} : \forall v \in \mathcal{V}\}$ into latent representation $\boldsymbol{E}_0$: $\{\boldsymbol{e}_0^{(v)} : \forall v \in \mathcal{V}\}$ with the ReLU activation function (line 1). $\boldsymbol{W}_T$ and $\boldsymbol{b}_T$ are the trainable weight and bias parameters. The transformer can help obtain high-level node attributes with sufficient representations. In the second step of node embedding, given the transformed cell embedding $\boldsymbol{e}_0$, deep learning GAT layers are then employed to aggregate the representation from neighborhoods (called aggregators) and to encode nodes attributes (called encoders). In our work, an aggregator and an encoder can be performed in a GAT layer. Aggregators gather the attribute information from the node's neighbors via attention coefficients $\boldsymbol{\alpha}$, which indicate the importance of neighborhood attributes to the target node. And encoders are applied to achieve nonlinear transformation by a weight matrix and activation function. The embedding process is performed multiple times for collecting more structural information. After the final embedding finishing, the structure decoder takes the result $\boldsymbol{E}$ as input and decodes it to reconstruct the original network structure.

Suppose that all the parameters in GAT layers are obtained after training. The progress of node embedding is concluded in Algorithm 1. Based on the given attributed network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$, Algorithm 1 takes the node attributes $\boldsymbol{X}$: $\{\boldsymbol{x}^{(v)}:\forall v \in \mathcal{V}\}$ defined in Section III-A as inputs, and output the node embeddings $\boldsymbol{E}$: $\{\boldsymbol{e}^{(v)}:\forall v \in \mathcal{V}\}$. Since the node embedding is expected to aggregate the information from multiorder neighbors for getting more accurate representations, a search depth $D$ is specified to indicate the neighborhood region range (i.e., defined as hop).

In the aggregator, it takes the representations of target node $v$ and nodes in the neighbor set $\mathcal{N}_v$ generated in the $(d-1)$th iteration as input. Then it generates a new representation for node $v$ denoted by $\boldsymbol{g}_d^{(v)}$ based on attention coefficients (line 5). The attention coefficients are obtained via feed forward neural networks as shown in Fig. 7. They are further normalized via the softmax function to make coefficients easily comparable across different neighbors. For instance, we can calculate attention coefficients $\boldsymbol{\alpha}_d^{(ij)}$ as follows:

$$\boldsymbol{\alpha}_d^{(ij)} = \frac{\exp\Big(\text{LeakyReLU}\Big((\boldsymbol{a}_d)^\top\Big[\boldsymbol{e}_d^{(i)} \| \boldsymbol{e}_d^{(j)}\Big]\Big)\Big)}{\sum_{f \in \mathcal{N}_{(i)}} \exp\Big(\text{LeakyReLU}\Big((\boldsymbol{a}_d)^\top\Big[\boldsymbol{e}_d^{(i)} \| \boldsymbol{e}_d^{(f)}\Big]\Big)\Big)} \tag{1}$$

where $i$ is the target node and the $j$ is its neighbor belongs to neighborhood set $N_i$. $\cdot^\top$ represents transposition and $\|$ is the concatenation operation. $\boldsymbol{a}_d \in \mathbb{R}^{2K_d}$ is the trainable weight vector. The negative input slope of LeakyReLU nonlinear function is set as 0.2.

In the encoder, a nonlinear transformation is performed to encode the aggregated representation with a weight matrix $\boldsymbol{W}_d \in \mathbb{R}^{K_d \times K_{d-1}}$ and an activation function (ReLU). The example shown in Fig. 7 illustrates the procedure of achieving node embedding when $D = 2$. Obviously, the node embedding output $\boldsymbol{E}$ is combined with the information of $d$-hop neighbors after $d$ iterations.

*Structure Decoder:* Once the maximum depth $D$ is reached, we can obtain the embedding feature matrix $\boldsymbol{E}$ after aggregating the neighbor node attributes. Structure decoder takes it as inputs to reconstruct the original network structure to get estimated adjacency matrix $\widehat{\boldsymbol{A}}$: $\{\boldsymbol{e}_D^{(v)} : \forall v \in \mathcal{V}\}$, as shown in 2

$$\widehat{\boldsymbol{A}} = \text{sigmoid}\Big(\boldsymbol{E}\boldsymbol{E}^\top\Big). \tag{2}$$

Given a certain node, if the connectivity values can be approximated through the structure reconstruction decoder, it means the node is normal with a high probability from the perspective of structure information. Otherwise, the connectivity values cannot be well reconstructed, which implies that its structure information does not conform to the patterns of the major normal nodes, and the node may be anomalous with high probability. Therefore, we use the structure reconstruction error $\boldsymbol{R}_A = \boldsymbol{A} - \widehat{\boldsymbol{A}}$ to indicate the probability of anomaly with respect to network structure for critical cells detection.

## C. Circuit Attribute Deep Autoencoder

The circuit attribute deep autoencoder contains an attribute encoder and an attribute decoder. In encoding progress, it encodes cell attribute information $\boldsymbol{X}$ into new embedding $\boldsymbol{Z}$, while the decoder tries to reconstruct the original cell attributes based on the latent embedding $\boldsymbol{Z}$. Finally, the reconstruction

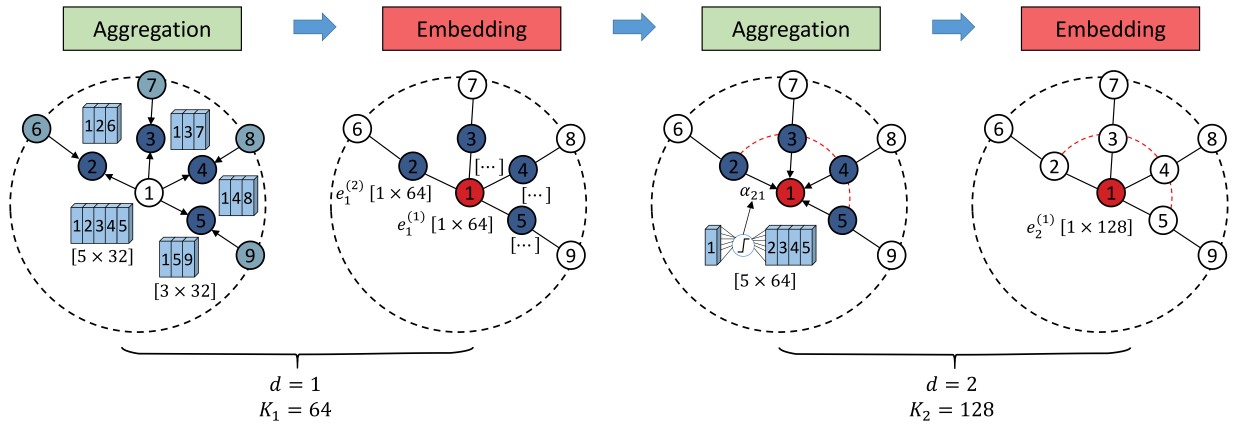


Fig. 7. Illustration to compute the node embedding for node 1 with GAT-layers when $D$ equals 2. And the procedure to compute the attention coefficient between node 1 and node 2 based on a single-layer feedforward neural network. $K_d$ is the dimension of the embedding results the $d$th GAT layer and $K_0 = 32$ in the feature transform layer.

error $\boldsymbol{R}_X$ for each cell from the circuit attribute perspective can be generated.

*Attribute Encoder:* In the attribute encoder, we use two nonlinear transform layers to get attribute embeddings $\boldsymbol{Z}$ based on the transposed attribute matrix $\boldsymbol{X}^\top$. It helps improve critical cell detection accuracy. The progress can be formulated as follows:

$$\boldsymbol{Z}_0 = \sigma\left((\boldsymbol{X})^\top \boldsymbol{W}_A^{(1)} + \boldsymbol{b}_A^{(1)}\right) \tag{3}$$

$$\boldsymbol{Z} = \boldsymbol{Z}_0 \boldsymbol{W}_A^{(2)} + \boldsymbol{b}_A^{(2)} \tag{4}$$

where $\boldsymbol{W}_A^{(1)}$ and $\boldsymbol{W}_A^{(2)}$ are the learnable weights in the two nonlinear transforming layers; and $\boldsymbol{b}_A^{(1)}$ and $\boldsymbol{b}_A^{(2)}$ are the learnable bias. After training, the weights and bias can be used for unseen designs directly without retraining in our work.

*Attribute Decoder:* The inputs of the attribute decoder are the node embeddings $\boldsymbol{E}$ and the attribute embeddings $\boldsymbol{Z}$, which are generated through the structure encoder and the attribute encoder, respectively. Similar to the structure decoder, the attribute decoder is used to decode the original node attribute. Interactions between network structure and node attribute are jointly captured. Note that different from the structure decoder, no activation function is utilized in the attribute decoder for the arbitrary-valued attribute

$$\widehat{\boldsymbol{X}} = \boldsymbol{E}\boldsymbol{Z}^\top. \tag{5}$$

Then with the attribute reconstruction error matrix $\boldsymbol{R}_X = \boldsymbol{X} - \widehat{\boldsymbol{X}}$, we can predict anomalies in the attributed networks from the attribute perspective. It can help improve critical cell detection accuracy in our work.

## D. Critical Cell Detection Classifier

We define the anomaly score $\boldsymbol{S} : \{s^{(v)}{:}\forall v \in \mathcal{V}\}$ of each node as the reconstruction error from both network structure and node attribute perspective

$$\boldsymbol{S} = (1-\beta)\left\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\right\|_2 + \beta\left\|\boldsymbol{X} - \widehat{\boldsymbol{X}}\right\|_2 \tag{6}$$

where $\beta$ is the parameter that tradeoffs structure reconstruction and attribute reconstruction. And it is defined as 0.5 in our work. After computing all anomaly scores of nodes in given circuits, the minimum score among anomalies is regarded as a threshold number at first. Then all the anomaly scores need to be normalized with it, and the results are $s_N^{(v)}\ \forall v \in \mathcal{V}$. In the critical cell detection classifier, the sigmoid function is used to estimate the probability of a cell belonging to critical or uncritical cells

$$\hat{y}^{(v)} = \frac{1}{1 + e^{-s_N^{(v)}}}\ \forall v \in \mathcal{V}. \tag{7}$$

## E. Training Progress

For graph learning (including circuit structure autoencoder and circuit attribute autoencoder), uncritical cells are used while in the training progress. The training objective of graph learning is to minimize the reconstruction errors of both network structure and node attribute for uncritical cells. After achieving the objective, uncritical and critical cells can be distinguished and classified based on the anomaly score $s^{(v)}$. And the loss function of graph learning is defined as follows, where $\mathcal{V}_{un}$ is the uncritical cells set:

$$Loss_g = \sum_{v \in \mathcal{V}_{un}} s^{(v)}. \tag{8}$$

For the critical cell detection classifier, all cells (critical cells and uncritical cells) are used in the training progress. The training objective of a critical cell detection classifier is to maximize the reconstruction errors of critical cells. After achieving the objective, the anomaly scores of critical cells (anomalies) can be much larger than uncritical cells (normal nodes). The target is achieved by using a supervised logistic classifier in our work. Then, the loss function of the classifier is defined as follows:

$$Loss_c = -\frac{1}{n}\sum_{v \in \mathcal{V}} y^{(v)} \log \hat{y}^{(v)} + \left(1 - y^{(v)}\right) \log\left(1 - \hat{y}^{(v)}\right) \tag{9}$$

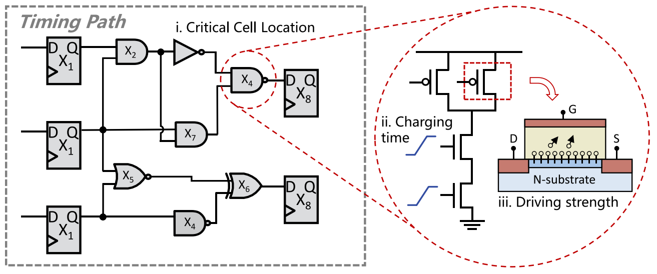


Fig. 8. Illustration of PC computation.

where $y^{(v)}$ is the label of each cell, which is obtained from aging-aware SPICE flow (as shown in Fig. 2). $\hat{y}^{(v)}$ is the probability value of each, which is obtained from our critical cell detection classifier. $n$ is the number of nodes in $\mathcal{V}$. Finally, the overall loss function in the training progress of our GAT-based work is defined as follows:

$$Loss_{all} = Loss_g + Loss_c. \tag{10}$$

The Adam [34] algorithm is utilized for optimization with a 0.001 learning rate. The overall proposed model is trained in an end-to-end fashion, which means all the parameters in the network can be trained without any manual assistance.

## IV. Path Criticality Computation

After a multistage GAT model is trained, it can identify critical cells in a netlist. However, the aging-induced delay of cells imposes different effects on the path delay even though all of them are critical cells. In order to select CPs accurately, we must define the detailed numerical relationship between critical cells and paths. In this section, we proposed an algorithm to identify which critical cells significantly degrade path timing performance and calculate PC. According to our observations, the critical cell with different types, driving strengths, and locations has different influences on the PC.

As shown in Fig. 8, we can directly get the related cross-layer parameters by analyzing the circuit information. The location information $L$ can be collected from cell-level timing results, which indicates the cell positions in the timing paths. The output load is obviously different for cells located in the fan-in-cones, which further affects the cell delay.

Moreover, we use the number of transistors $T$ in the target cell to represent the feature of cell types. The cell types contain DFF, BUFF, NOT, AND, NAND, OR, NOR, XOR, and XNOR with multi-inputs. According to the physical principles, the cell delay can be defined as the charging and discharging time of transistors in the circuit. Thus, the size of transistors applied in the cell can represent the feature of driving strength $R$. These values can be directly obtained from the standard cell library. All these data must be normalized to be $L_N$, $T_N$, and $R_N$. The PC of path $j$ can be calculated as follows:

$$PC_j = \sum_{i \in N_j} \left( L_N^{(i)} + T_N^{(i)} + R_N^{(i)} \right) \cdot \hat{y}^{(i)} + S_N^{(j)} \tag{11}$$

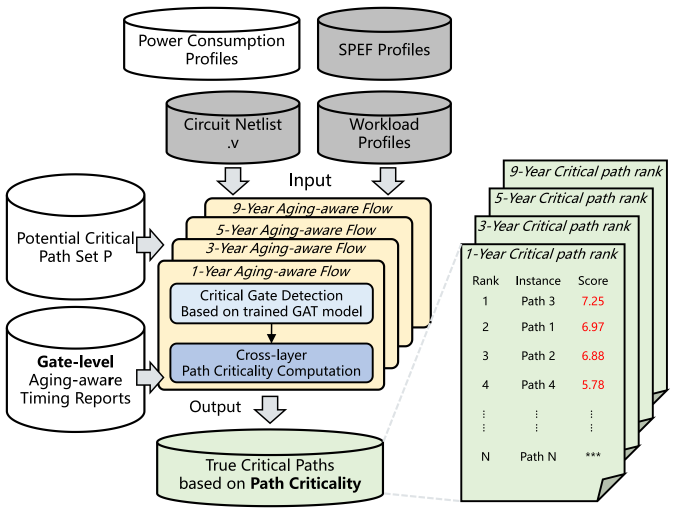

Power Consumption Profiles
SPEF Profiles
Circuit Netlist .v
Workload Profiles
Input
Potential Critical Path Set P
Gate-level Aging-aware Timing Reports
9-Year Aging-aware Flow
5-Year Aging-aware Flow
3-Year Aging-aware Flow
1-Year Aging-aware Flow
Critical Gate Detection Based on trained GAT model
Cross-layer Path Criticality Computation
Output
True Critical Paths based on Path Criticality
9-Year Critical path rank
5-Year Critical path rank
3-Year Critical path rank
1-Year Critical path rank
Rank Instance Score
1 Path 3 7.25
2 Path 1 6.97
3 Path 2 6.88
4 Path 4 5.78
N Path N ***

Fig. 9. Proposed parallel flow of CP selection. On multiple GPUs, we achieve graph learning in parallel to get a trained model under different aging conditions. And compute the PC on a multicore machine. Finally, a CP can be selected based on the path rank list.

where $N_j$ is the cell set of path $j$. $\hat{y}^{(i)}$ is the result of GAT-based critical cell detection which means whether the cell is critical or uncritical for aging-aware timing analysis. $S_N^{(j)}$ is the normalized path slack of path $j$.

As an output, the CP ranking lists after aging can be obtained via the proposed algorithm. Furthermore, the designer can use the ranking list to improve the circuit performance and reliability more efficiently by focusing on true CPs.

## V. Aging-Aware Critical Path Selection

The overall aging-aware CP selection flow is proposed in Fig. 9 with four different aging conditions, including 1-year, 3-year, 5-year, and 9-year. Given the circuit netlist, SPEF profiles, technology profiles, power consumption profiles, potential CP sets, and cell-level aging-aware timing results, our flow can generate accurate critical timing paths without running aging-aware SPICE. The cell-level aging-aware timing results and potential CP sets are generated by aging-aware STA using PrimeTime [10] and PrimeLib [25]. The workload profiles are generated through workload analysis using logic simulator Modelsim [31]. Additionally, the power consumption profiles are generated via Design Compiler [32]. The SP profiles are computed based on workload profiles and temperature profiles are computed based on power consumption profiles.

First, we take the initial attributes defined in Section III-A as inputs. Then the GAT-based trained model in Section III can help us to achieve critical cell detection considering aging effects based on cell attributes and circuit structures. The PC of all the potential CPs can be computed by our proposed algorithm in Section IV. According to PC ranking lists, we can select the CPs under different requirements. Thus, the problems of aging-aware STA proposed in Section II-C can be solved by our advanced aging-aware CP selection flow, including a GAT-based trained model for critical cells detection and a practical PC computation, demonstrated in Fig. 9.

Different from conventional machine learning methods which just focus on cell features, the graph learning framework in our work can aggregate the circuit structural information and cell features to generate a better representation for each cell in the circuit. The new representation helps detect critical cells on CPs effectively and accurately, which is because the influence of aging effects on the target cell and neighbor cells are all considered. We define PC to quantify the detailed numerical effect of critical cells on path delay after aging considering both transistor-level process parameters and circuit-level information. There is an obvious improvement in CP selection accuracy based on the PC rather than cell-level timing results according to the experiments on industrial designs.

In our flow, there are multiple pretrained critical cell detection models. However, the training process is time consuming, which limits the efficiency of our GAT-based framework. It is necessary to leverage a parallel training scheme with multiple GPUs. In our training scheme, we can parallelize it by partitioning the chunk-based dataflow over multi-GPUs. Each GPU processes one graph in our parallel framework with a complete and dependent adjacency matrix and node representation matrix. Finally, the main thread gathers all these results, calculates the loss, and then does backpropagation to update the model. Besides, our parallel PC computation is based on main-sub threads architectures achieved with Pthreads. There is only one main thread that can create multiple sub threads and manages the shared memory. The sub thread is used to collect related parameters, including cell locations, cell types, and transistor sizes, and compute the PC for timing paths. Benefiting from the parallel technology, all the process of critical cell detection and PC computation can be performed on multiple GPUs and CPUs to achieve speedup on the overall flow.

## VI. Experimental Results

The experiments are performed on five unseen industrial designs implemented in 16-nm technology. The potential CP set is generated via aging-aware STA based on our settings. The false paths, including timing paths across asynchronous clock domains, are removed. Statistics of designs used in our experiments are summarized in Table II, where #Cells #10%-Ps, and #CCs indicate the number of cells, top-10% CPs, and critical cells on these paths in the netlist, respectively. Note that the ground truths of top-10% CPs and critical cells are obtained from the transistor-level aging-aware results of paths in potential CP set. The results are generated via aging-aware SPICE using HSPICE [20]. The GAT is implemented with PyTorch and trained on a Linux machine with 32 cores and 4 NVIDIA Tesla V100 GPUs in parallel. The total memory used in training is 128 GB. The benchmark circuits used for training and testing the GAT-based model contain all ISCAS'89, IWLS'05 circuits, and other renowned industrial designs synthesized with TSMC16nm technology. The details of these designs used for training are listed in Table II. As discussed in Section III, search depth is a crucial parameter that affects

TABLE II
Training and Testing Benchmark Information

| Circuit | #Edges | #Cells | #10%-Ps | # CCs | #CCs/Cells | Usage |
|---|---|---|---|---|---|---|
| s35932 | 20563 | 16065 | 1954 | 2096 | 13.05% | Training |
| s38584 | 22674 | 19253 | 1694 | 2193 | 11.39% | |
| s38417 | 23651 | 22179 | 1779 | 3792 | 17.10% | |
| b14 | 24893 | 21680 | 1987 | 2569 | 11.85% | |
| b15 | 25130 | 20186 | 2265 | 2659 | 13.17% | |
| b17 | 84571 | 61044 | 4743 | 5194 | 8.51% | |
| b20 | 45639 | 31258 | 2703 | 3098 | 9.91% | |
| b22 | 51630 | 39385 | 3193 | 3690 | 9.37% | |
| ethernet | 91237 | 72609 | 8964 | 9154 | 12.61% | |
| des3 | 10657 | 86309 | 7365 | 7693 | 8.91% | |
| vga | 140325 | 115011 | 9863 | 12065 | 10.49% | |
| des | 176394 | 135555 | 10654 | 13098 | 9.66% | |
| jpeg | 231934 | 219064 | 15657 | 18609 | 8.49% | |
| D1 | 82197 | 81792 | 5973 | 6838 | 9.83% | Testing |
| D2 | 85058 | 84127 | 4219 | 5305 | 6.31% | |
| D3 | 93812 | 90859 | 2651 | 3961 | 4.36% | |
| D4 | 90905 | 113168 | 5792 | 6270 | 5.54% | |
| D5 | 138171 | 136537 | 5496 | 6745 | 4.94% | |
| b1 | 1593 | 1325 | 187 | 262 | 19.77% | |
| b2 | 3517 | 2068 | 375 | 439 | 21.23% | |
| b3 | 3437 | 2964 | 441 | 462 | 15.59% | |
| b4 | 5239 | 4561 | 618 | 725 | 15.79% | |

TABLE III
CP Selection Accuracy (%) Based on the Potential CP Set When the Baseline Results Are Generated Through Aging-Aware SPICE Based on All Paths

| Des. | Critical path selection accuracy (%) | | | | | |
|---|---|---|---|---|---|---|
| | 1-year | | | 9-year | | |
| | 1% | 5% | 10% | 1% | 5% | 10% |
| b1 | 94.65 | 96.79 | 99.46 | 94.65 | 95.72 | 98.39 |
| b2 | 97.33 | 98.4 | 99.73 | 94.67 | 96.80 | 99.47 |
| b3 | 97.73 | 99.09 | 99.55 | 95.46 | 98.19 | 99.32 |
| b4 | 96.76 | 98.38 | 99.19 | 95.15 | 97.41 | 98.87 |
| ave. | 96.62 | 98.16 | 99.48 | 94.98 | 97.03 | 99.01 |

the performance of a GAT learning framework. According to the experimental results of detection accuracy after learning for 300 epochs, the search depth is set to 3 for all experiments. Furthermore, we can achieve top-5% and top-1% CP selection by fine-tuning our labeled critical cells in the training set. For example, if the top-1% paths are required to be selected, the cells on the top-1% paths after aging-aware SPICE in the training set will be regarded as critical cells. After training, the GAT-based critical cell detection method can figure out cells on aging-aware top-1% paths without aging-aware SPICE for unseen designs.

### A. *Efficiency of Potential Critical Path Set*

We use PrimeTime to generate the potential CP set $\mathcal{P}$ for each design, utilizing our specified settings: nworst 10 on endpoints and nworst 1 through cells. To test the effectiveness of our approach, we use aging-aware SPICE to generate ground truths on a path set consisting of almost all paths, with nworst 10 000 at each endpoint (the number of exacted timing paths at each endpoint being less than 10 000). We then compare the accuracy of CP selection using our method based on the original path set $\mathcal{P}$ for small designs with ground truths. Statistics of these small designs used in the experiments are summarized in Table II. The results are displayed in Table III. For instance, we generate the top 10% path set for the b1

TABLE IV
CRITICAL CELL DETECTION ACCURACY COMPARISON

| Design | LOF | SCAN | AMEN | Radar | Anom. | Dom. | GAT |
|---|---|---|---|---|---|---|---|
| D1 | 0.492 | 0.378 | 0.498 | 0.715 | 0.796 | 0.847 | **0.983** |
| D2 | 0.444 | 0.298 | 0.472 | 0.692 | 0.807 | 0.884 | **0.991** |
| D3 | 0.432 | 0.342 | 0.543 | 0.703 | 0.812 | 0.821 | **0.994** |
| D4 | 0.427 | 0.272 | 0.505 | 0.687 | 0.782 | 0.804 | **0.987** |
| D5 | 0.462 | 0.312 | 0.433 | 0.699 | 0.736 | 0.753 | **0.992** |
| ave. | 0.451 | 0.320 | 0.490 | 0.699 | 0.787 | 0.822 | **0.989** |

design using the potential path set. Next, we compare this path set with the top 187 paths generated by aging-aware SPICE using almost all timing paths of the b1 design. According to the results, the potential CP set is reasonable, demonstrating its potential in achieving true aging-aware CP selection.

## B. Accuracy of Critical Cell Detection

We compare the critical cell detection performance of the GAT-based framework with the state-of-the-art machine learning methods, including LOF [35], SCAN [36], AMEN [37], Radar [38], Anomalous [39], and Dominant [40]. Each time we use open-source designs in Table II for training and the industrial designs for testing while using different machine-learning methods. For head-to-head comparisons, the objectives of our work and other machine learning methods are the same for maximizing the reconstruction errors of critical cells. The ground truths of critical cells after aging are selected through aging-aware SPICE shown in Fig. 2. The experimental results in Table IV show that the proposed framework significantly outperforms all baselines.

For the D1 circuit, our framework outperforms LOF [35] by 49.1%, SCAN [36] by 60.5%, and AMEN [37] by 48.5% because LOF and SCAN consider only network structure or node attribute. AMEN [37] is designed for anomalous neighborhoods rather than the node itself, so the method's detection accuracy cannot meet the requirement for the current reliability-aware circuit design flow. Besides, the residual analysis [38] and cur matrix decompositions-based method [39] are not sensitive to network sparsity. They meet bottlenecks for large-scale designs with limited learning ability. Thus, our framework increases the detection accuracy by 29.3% compared with Radar [38] and 25.6% compared with Anomalous [39] for the largest design D5. Compared with more recent Dominant [40], our work achieves gains by 13.6%, 10.9%, % 17.3%, and 23.9% on the five industrial designs, respectively. Compared with the single structure encoder used in Dominat, two separate encoders, including circuit structure and attribute encoder, are proposed in our work. They can jointly achieve node embedding and attribute embedding, which considers the complex interactions between network structure and node attribute. The improvement helps achieve higher detection accuracy.

In addition, our GAT-based model is tested under different workloads and the results are shown in Table V. The different workloads in these experiments include random input vectors with average SP equaling 0.5, 0.6, and 0.7. Similar methods to obtain workloads are used in [24] and [30]. From the results,

TABLE V
CRITICAL CELL DETECTION ACCURACY OF OUR GAT-BASED METHOD UNDER DIFFERENT WORKLOADS AND THE STANDARD DEVIATION ($\sigma$) OF THEM

| Design | SP=0.5 | SP=0.6 | SP=0.7 | standard dev. ($\sigma$) |
|---|---|---|---|---|
| D1 | 0.983 | 0.979 | 0.988 | 6.38E-03 |
| D2 | 0.991 | 0.986 | 0.993 | 5.10E-03 |
| D3 | 0.994 | 0.992 | 0.983 | 8.29E-03 |
| D4 | 0.987 | 0.991 | 0.982 | 6.38E-03 |
| D5 | 0.992 | 0.993 | 0.996 | 2.94E-03 |
| ave. | 0.989 | 0.988 | 0.988 | 8.16E-04 |

it is obvious that our GAT-based critical cell detection method can achieve high detection accuracy under different workloads, which reaches 0.989, 0.988, and 0.988 under three kinds of workloads. Moreover, the small values of standard deviation ($\sigma$) under different workloads illustrate that our work can obtain accurate critical cells stably. Obviously, the GAT-based model can obtain significantly stronger performance in distinguishing between critical and uncritical nodes than classical learning models. The model training progress is a little timing-consuming, with about 12 h on a single GPU. However, the parallel training method on multiple GPUs can help us achieve a 6× speedup on our servers.

## C. Performance of Critical Path Selection

Using the results of aging-aware SPICE as ground truths, Tables VI–VIII show the CP selection accuracy results of our work, the method used in ICCAD13 [24] and aging-aware STA under different aging and workload conditions. Fig. 10 shows the runtime cost comparison. The aging-aware SPICE is achieved using HSPICE [20] and MOSRA aging model [7]. The aging-aware STA is achieved using the commercial STA tool (PrimeTime) [10] and the aging-aware timing library cell characterizes tool (PrimeLib) [25].

The aging-induced timing degradation of circuits is workload-dependent. For illustrating the efficacy of our framework under different workloads, our method is tested under different workload profiles when the average SP equals 0.5, 0.6, and 0.7. Based on (12), we compute the average delay error of wrong CPs selected via aging-aware STA, ICCAD13 method [24], and our work (DELAY_EOR$_{asta}$, DELAY_EOR$_{icc}$ and DELAY_EOR$_{\text{ours}}$)

$$\text{DELAY\_EOR}_{.} = \frac{\sum_{p_1 \in \mathcal{P}_{\text{spice}}} D_{p_1} - \sum_{p_2 \in \mathcal{P}_{.}} D_{p_2}}{N_{\mathcal{P}_{.}^{\text{wrong}}}} \quad (12)$$

where $\mathcal{P}_{\text{spice}}$ represent the CP set generated by aging-aware SPICE. $\mathcal{P}_{.}$ represent the CP set generated by ICCAD13 method [24], aging-aware STA, or our work. $D_p$ represents the real delay of timing path $p$ generated by aging-aware SPICE. $N_{\mathcal{P}_{.}^{\text{wrong}}}$ represent the number of wrong paths in CP set generated via aging-aware STA, ICCAD13 method [24], or our work, compared with aging-aware SPICE. The results of DELAY_EOR$_{.}$ are listed in Tables IX and XI where a smaller value means better performance.

*Accuracy Under Different Aging Conditions:* Under a 1-year aging condition, the average accuracy of aging-aware STA can

TABLE VI
CP SELECTION ACCURACY (%) COMPARISON (AVERAGE SP EQUALS 0.5)

| Design | 1-year | | | | | | | | | 9-year | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Aging-aware STA | | | ICCAD13 [24] | | | ours | | | Aging-aware STA | | | ICCAD13 [24] | | | ours | | |
| | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% |
| D1 | 89.12 | 93.67 | 94.22 | 93.47 | 95.45 | 97.29 | 97.99 | 98.96 | 99.84 | 81.42 | 85.33 | 86.77 | 90.79 | 92.77 | 95.66 | 96.99 | 98.35 | 99.74 |
| D2 | 87.20 | 91.28 | 92.53 | 90.28 | 93.84 | 94.86 | 98.81 | 99.57 | 99.54 | 77.72 | 82.08 | 85.66 | 87.67 | 90.42 | 93.15 | 97.16 | 99.29 | 99.72 |
| D3 | 88.31 | 92.76 | 93.21 | 94.34 | 95.85 | 97.89 | 98.49 | 98.86 | 99.62 | 78.12 | 83.25 | 86.23 | 88.68 | 92.61 | 95.66 | 97.74 | 98.64 | 99.32 |
| D4 | 86.88 | 90.06 | 91.97 | 90.16 | 95.72 | 96.58 | 98.96 | 99.14 | 99.87 | 76.35 | 81.42 | 84.96 | 86.02 | 91.13 | 94.56 | 97.58 | 98.41 | 99.45 |
| D5 | 85.99 | 89.85 | 91.34 | 91.08 | 93.96 | 94.74 | 97.82 | 98.58 | 99.69 | 72.34 | 78.02 | 84.21 | 87.81 | 90.21 | 93.10 | 97.09 | 98.14 | 99.47 |
| ave. | 87.50 | 91.52 | 92.66 | 91.87 | 94.96 | 96.27 | 98.41 | 99.02 | 99.72 | 77.19 | 82.02 | 85.57 | 88.19 | 91.43 | 94.43 | 97.31 | 98.57 | 99.54 |

TABLE VII
CP SELECTION ACCURACY (%) COMPARISON (AVERAGE SP EQUALS 0.6)

| Design | 1-year | | | | | | | | | 9-year | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Aging-aware STA | | | ICCAD13 [24] | | | ours | | | Aging-aware STA | | | ICCAD13 [24] | | | ours | | |
| | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% |
| D1 | 85.60 | 89.42 | 90.11 | 89.95 | 92.47 | 94.56 | 97.66 | 98.89 | 99.78 | 79.07 | 81.42 | 83.32 | 86.44 | 88.98 | 93.86 | 96.65 | 98.56 | 99.65 |
| D2 | 84.83 | 88.53 | 90.47 | 88.15 | 93.84 | 92.87 | 98.58 | 99.29 | 99.62 | 74.40 | 78.24 | 82.18 | 84.59 | 87.63 | 92.25 | 96.92 | 98.58 | 99.27 |
| D3 | 84.16 | 87.48 | 91.06 | 88.31 | 94.57 | 95.51 | 98.11 | 99.17 | 99.36 | 75.10 | 80.08 | 83.97 | 83.03 | 88.91 | 91.47 | 96.61 | 98.42 | 99.21 |
| D4 | 83.25 | 86.98 | 89.14 | 87.05 | 95.72 | 91.76 | 98.62 | 99.27 | 99.91 | 70.99 | 74.93 | 78.50 | 82.39 | 87.09 | 89.33 | 97.41 | 99.17 | 99.41 |
| D5 | 83.08 | 85.77 | 88.99 | 85.63 | 93.96 | 92.23 | 97.63 | 98.36 | 99.29 | 67.98 | 71.11 | 77.18 | 82.35 | 88.79 | 90.74 | 96.72 | 97.67 | 99.58 |
| ave. | 84.18 | 87.64 | 89.95 | 87.82 | 94.11 | 93.39 | 98.12 | 98.99 | 99.59 | 73.51 | 77.16 | 81.03 | 83.76 | 88.28 | 91.53 | 96.86 | 98.47 | 99.42 |

TABLE VIII
CP SELECTION ACCURACY (%) COMPARISON (AVERAGE SP EQUALS 0.7)

| Design | 1-year | | | | | | | | | 9-year | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Aging-aware STA | | | ICCAD13 [24] | | | ours | | | Aging-aware STA | | | ICCAD13 [24] | | | ours | | |
| | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% |
| D1 | 82.92 | 86.20 | 88.25 | 84.76 | 87.68 | 93.27 | 98.49 | 99.29 | 99.49 | 74.05 | 78.27 | 81.08 | 82.92 | 84.70 | 91.09 | 96.48 | 98.82 | 99.34 |
| D2 | 81.51 | 85.97 | 89.33 | 85.54 | 89.62 | 91.68 | 98.10 | 99.05 | 99.74 | 72.03 | 77.91 | 80.92 | 80.56 | 84.88 | 90.09 | 97.16 | 98.53 | 99.57 |
| D3 | 83.03 | 86.65 | 90.08 | 82.65 | 89.59 | 92.38 | 96.98 | 98.64 | 99.32 | 71.33 | 79.03 | 83.06 | 80.01 | 83.48 | 91.47 | 96.22 | 98.11 | 99.47 |
| D4 | 80.49 | 82.56 | 85.93 | 83.60 | 91.78 | 90.18 | 98.27 | 99.17 | 99.71 | 67.54 | 72.65 | 76.61 | 79.63 | 84.43 | 89.33 | 97.06 | 98.54 | 99.52 |
| D5 | 81.62 | 84.02 | 87.30 | 84.90 | 89.30 | 91.16 | 97.09 | 98.72 | 99.64 | 63.97 | 70.96 | 73.49 | 80.71 | 86.14 | 90.74 | 96.36 | 98.11 | 99.43 |
| ave. | 81.91 | 85.08 | 88.18 | 84.29 | 89.59 | 91.73 | 97.79 | 98.96 | 99.58 | 69.79 | 75.76 | 79.03 | 80.77 | 84.72 | 90.55 | 96.66 | 98.42 | 99.46 |

reach 92.66 However, it drops with the value of K% from 10% (92.66%) to 5% (91.55%) and 1% (87.50%). Then, our framework can achieve an obvious accuracy improvement. Under a 9-year condition, it can be seen that the average accuracy of aging-aware STA is just 77.19%, 82.02%, and 85.57% on top-1%, 5%, and 10% path set. It indicates that the aging-aware STA cannot predict aging-induced timing degradation on timing paths accurately under serious aging conditions. Compared with the poor performance of aging-aware STA, the average accuracy of our framework can achieve 96.25%, 98.12%, and 99.43% as shown in Table VI, which can meet the high-reliability requirement.

*Accuracy Under Different Workload Conditions:* From the results shown in Tables VI–VIII, it is obvious that the average accuracy of aging-aware STA drops with the increment of the average SP. When the average SP equals 0.5, the accuracy of top-K% path sets generated by aging-aware STA reach 87.50%, 91.52%, and 92.66% under 1-year aging condition. However, the accuracy of the top-1% path set generated by aging-aware STA drops to 81.91% when the average SP equals 0.7. Compared with inaccurate aging-aware STA, the accuracy of top-K% path sets generated by our work can achieve 97.79%, 98.96%, and 99.58% as shown in Table VIII. It means our work can generate accurate CPs under different workloads.

*Delay Error of Wrong CPs:* As shown in Table IX, it is obvious that DELAY_EOR$_{asta}$ is much larger than DELAY_EOR$_{\text{ours}}$. In addition, the results of aging-aware STA increase significantly under more serious aging conditions. However, the results of our work can always keep at a very low level. For wrong CPs selected by aging-aware STA, the average delay error on five industrial circuits under 9-year aging conditions reaches 50.81, 43.40, and 32.02 ps in the top 1%, 5%, and 10% path sets. However, the average error of wrong CPs selected by our work reduces to 1.96, 1.15, and 0.25 ps. The large delay error on wrong CPs selected via aging-aware STA brings overdesign on these wrong paths, which degrades the circuit timing performance and improves the power consumption significantly. On the other hand, the large delay error causes low timing yield on final tape-out circuits, because the delay of unselected true CPs is considered optimistically without fixing the timing violations.

*Runtime:* As shown in Fig. 10, the runtime cost of our work is 75.58 s on average for different designs scaling from 80k to 130k cells. The method used in ICCAD13 [24] can achieve more accurate selection than aging-aware STA, but the average runtime cost reaches 952.24 s which is much larger than aging-aware STA and our work. Compared with aging-aware SPICE using 32-threads, we can achieve an average 86.8× speedup using 1-thread on our benchmark circuits. Note that, we achieve parallel CP selection flow for multiple aging conditions (as shown in Fig. 9). The fast work is beneficial to improving design efficiency in the reliability-aware design flow. And there is nearly no increment in runtime under 1-year conditions and 9-year conditions, while traditional methods

TABLE IX
AVERAGE DELAY ERROR (PS) OF WRONG CPS SELECTED VIA AGING-AWARE STA, ICCAD13, AND OUR WORK

| Design | 1-year | | | | | | | | | 9-year | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DELAY_EOR$_{asta}$ (ps) | | | DELAY_EOR$_{icc}$ (ps) | | | DELAY_EOR$_{ours}$ (ps) | | | DELAY_EOR$_{asta}$ (ps) | | | DELAY_EOR$_{icc}$ (ps) | | | DELAY_EOR$_{ours}$ (ps) | | |
| | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% |
| D1 | 40.26 | 37.42 | 21.24 | 8.35 | 6.38 | 3.69 | 1.52 | 1.18 | 0.21 | 58.29 | 47.21 | 36.99 | 11.02 | 9.83 | 5.92 | 2.01 | 1.33 | 0.35 |
| D2 | 48.34 | 40.29 | 29.36 | 9.66 | 7.92 | 4.55 | 1.13 | 0.78 | 0.23 | 61.24 | 52.31 | 45.92 | 12.54 | 10.51 | 6.71 | 1.89 | 1.14 | 0.31 |
| D3 | 46.33 | 39.23 | 31.76 | 10.02 | 8.54 | 4.21 | 1.36 | 0.94 | 0.29 | 54.09 | 43.98 | 39.26 | 13.42 | 10.99 | 7.25 | 1.72 | 1.21 | 0.38 |
| D4 | 51.25 | 43.77 | 37.51 | 11.56 | 9.02 | 5.32 | 1.42 | 0.84 | 0.15 | 63.28 | 51.34 | 47.18 | 15.02 | 11.36 | 8.09 | 1.43 | 0.94 | 0.19 |
| D5 | 67.89 | 56.28 | 40.23 | 12.21 | 9.69 | 6.51 | 2.41 | 2.02 | 0.37 | 78.91 | 64.32 | 51.97 | 16.73 | 12.47 | 9.34 | 2.83 | 2.35 | 0.63 |
| ave. | 50.81 | 43.40 | 32.02 | 10.36 | 8.31 | 5.07 | **1.96** | **1.15** | **0.25** | 63.16 | 51.83 | 44.26 | 13.75 | 11.03 | 7.46 | **2.47** | **1.74** | **0.37** |

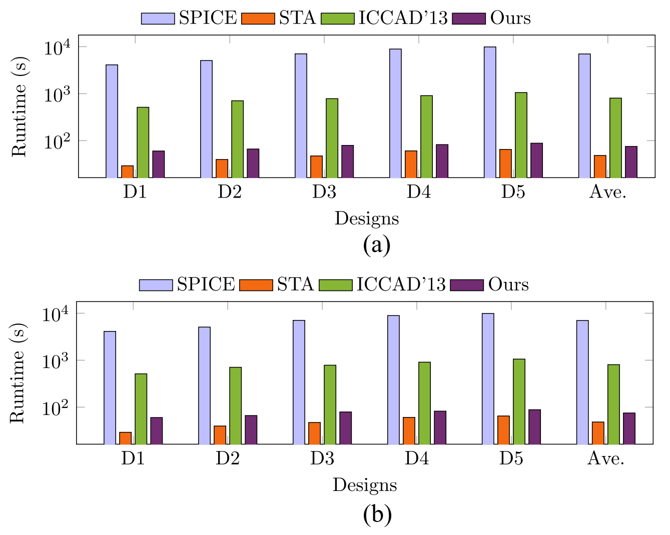


Fig. 10. Runtime comparison among aging-aware SPICE, aging-aware STA, and our work under (a) 1-year and (b) 9-year aging conditions.

need to analyze a more complex circuit with aging. Note that the runtime contains an aging-aware static STA based on PrimeTime, which means our work can be integrated into the current reliability design flow easily without more runtime overhead.

*Summary:* Compared with the ground truths generated via aging-aware SPICE, it is obvious that the average accuracy of aging-aware STA decreases when the aging effect is becoming serious. In the worst case (9-year aging condition and the average SP equals 0.7), the average accuracy of top-K% path sets generated by aging-aware STA reaches 69.81%, 75.77%, and 78.83% on five industrial designs. More importantly, the average delay error of wrong CPs is larger than 30 ps under a 1-year aging condition and 40 ps under a 9-year aging condition. However, the accuracy top-K% path sets generated by our work reaches 96.50%, 98.49%, and 99.49% as shown in Table VIII even in the worst case. Compared with the method proposed in [24] based on analyzing each transistor, the efficiency of our work is much higher. In summary, our work can achieve CP selection accurately and efficiently.

### *D. Performance on Designs With Timing Wall Problems*

Tables X and XI show the CP selection accuracy and average delay error result of our work, ICCAD13 [24] and aging-aware STA for designs with timing wall problems. Each

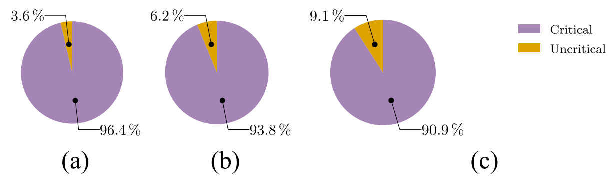


Fig. 11. Ratio of critical and uncritical cells on (a) Top-1%, (b) Top-5%, and (c) Top-10% paths.

testing design is resynthesized with a tight global timing constraint, and the average SP for workload conditions is 0.7. The accuracy of aging-aware STA for designs with timing wall problems surpasses that of designs synthesized with multiple timing constraints, with an average accuracy of 80.77% for top-1% CP selection under 9-year aging conditions. Nonetheless, our work achieves an average accuracy of 97.94%, significantly outperforming aging-aware STA. The average errors of wrong CPs selected by aging-aware STA are 27.18, 19.26, and 14.52 ps, while the results of our work reduce to mere 1.59, 1.00, and 0.25 ps.

### *E. Relationship Between Critical Cells and Paths*

When selecting CPs, the critical arcs play more important roles than critical cells. However, detecting critical arcs can be inefficient when runtime and memory usage are limited due to the exponential increase in the number of arcs as cells increase. Despite this, there is a strong connection between critical cells and paths, as evidenced by studies, such as [17], [18], and [19]. In fact, more than 90% of cells on top-10% paths are critical, as seen in Fig. 11. Therefore, it is valid to select CPs based on detecting critical cells.

## VII. CONCLUSION AND FURTHER WORK

This article introduces an aging-aware CP selection flow that includes a GAT-based critical cell detection framework and PC computation algorithm The end-to-end critical cell detection framework can distinguish the critical cells in the unseen circuits accurately considering aging effects. Then, the PC computation algorithm can generate PC rank lists based on the detection results and other cross-layer parameters. Compared with traditional machine learning methods, the proposed GAT model can achieve superior accuracy in critical cell detection. Experimental results show that the proposed flow can achieve high accuracy on industrial designs, while the aging-aware STA cannot meet. There will be two significant improvements in our further work: first, the combination of our current work

TABLE X
CP SELECTION ACCURACY (%) COMPARISON FOR DESIGNS WITH TIMING WALL PROBLEMS

| Design | 1-year | | | | | | | | | 9-year | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Aging-aware STA | | | ICCAD13 [24] | | | ours | | | Aging-aware STA | | | ICCAD13 [24] | | | ours | | |
| | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% |
| D1 | 86.77 | 90.19 | 92.99 | 89.95 | 92.47 | 98.18 | 99.16 | 99.60 | 99.88 | 81.08 | 84.30 | 86.97 | 86.44 | 88.98 | 93.86 | 99.00 | 99.26 | 99.80 |
| D2 | 86.02 | 91.04 | 92.60 | 88.15 | 93.84 | 92.87 | 99.29 | 99.48 | 99.81 | 80.80 | 84.64 | 87.94 | 84.59 | 87.63 | 92.25 | 98.34 | 98.77 | 99.62 |
| D3 | 88.31 | 90.34 | 94.27 | 88.31 | 94.57 | 95.51 | 98.49 | 99.62 | 99.74 | 81.52 | 85.44 | 88.65 | 83.03 | 88.91 | 91.47 | 96.61 | 98.94 | 99.59 |
| D4 | 85.67 | 88.98 | 91.51 | 87.05 | 95.72 | 91.76 | 99.14 | 99.55 | 99.97 | 80.49 | 83.94 | 86.38 | 82.39 | 87.09 | 89.33 | 97.93 | 99.27 | 99.86 |
| D5 | 85.63 | 89.56 | 92.74 | 85.63 | 93.96 | 92.23 | 98.73 | 99.24 | 99.78 | 79.99 | 82.06 | 85.41 | 82.35 | 88.79 | 90.74 | 97.82 | 98.65 | 99.71 |
| ave. | 86.48 | 90.02 | 92.82 | 87.82 | 94.11 | 94.11 | 98.96 | 99.50 | 99.84 | 80.77 | 84.08 | 87.07 | 83.76 | 88.28 | 91.53 | 97.94 | 98.98 | 99.72 |

TABLE XI
AVERAGE DELAY ERROR (PS) OF WRONG CPS SELECTED VIA DIFFERENT METHODS FOR DESIGNS WITH TIMING WALL PROBLEMS

| Design | 1-year | | | | | | | | | 9-year | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DELAY_EOR$_{asta}$ (ps) | | | DELAY_EOR$_{icc}$ (ps) | | | DELAY_EOR$_{ours}$ (ps) | | | DELAY_EOR$_{asta}$ (ps) | | | DELAY_EOR$_{icc}$ (ps) | | | DELAY_EOR$_{ours}$ (ps) | | |
| | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% | 1% | 5% | 10% |
| D1 | 15.19 | 10.24 | 6.39 | 6.22 | 4.98 | 2.54 | 1.43 | 0.93 | 0.14 | 23.24 | 15.63 | 10.18 | 8.91 | 6.02 | 3.92 | 1.71 | 1.02 | 0.27 |
| D2 | 17.17 | 12.25 | 7.21 | 6.41 | 5.12 | 2.91 | 0.92 | 0.48 | 0.17 | 25.53 | 17.09 | 12.36 | 10.02 | 6.89 | 3.63 | 1.30 | 0.61 | 0.22 |
| D3 | 18.51 | 13.64 | 8.76 | 6.89 | 4.21 | 3.02 | 1.17 | 0.76 | 0.21 | 26.37 | 18.21 | 14.51 | 11.43 | 7.42 | 4.59 | 1.29 | 0.87 | 0.25 |
| D4 | 21.39 | 15.87 | 9.63 | 7.04 | 5.48 | 3.28 | 0.94 | 0.63 | 0.08 | 28.34 | 20.36 | 16.29 | 12.31 | 8.67 | 5.21 | 1.23 | 0.71 | 0.13 |
| D5 | 25.17 | 20.61 | 10.92 | 7.19 | 6.37 | 4.02 | 2.05 | 1.52 | 0.22 | 32.42 | 25.13 | 19.27 | 13.85 | 9.91 | 6.02 | 2.43 | 1.79 | 0.38 |
| ave. | 19.49 | 14.52 | 8.58 | 6.75 | 5.23 | 3.15 | **1.30** | **0.86** | **0.16** | 27.18 | 19.26 | 14.52 | 11.30 | 7.78 | 4.67 | **1.59** | **1.00** | **0.25** |

with other false path identification techniques would result in a more automated approach to aging-aware CP selection. Second, efficient and accurate selection of CPs based on the identification of critical arcs can be achieved through sampling and transferring the learned information of arcs.

## REFERENCES

[1] R. Reis, Y. Cao, and G. Wirth, *Circuit Design for Reliability*. New York, NY, USA: Springer, 2015.

[2] S. X. Tan, M. Tahoori, T. Kim, S. Wang, Z. Sun, and S. Kiamehr, *Long-Term Reliability of Nanometer VLSI Systems: Modeling, Analysis and Optimization*. Cham, Switzerland: Springer, 2019.

[3] H. Amrouch, B. Khaleghi, A. Gerstlauer, and J. Henkel, "Reliability-aware design to suppress aging," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, 2016, pp. 1–6.

[4] S. Guo et al., "Towards reliability-aware circuit design in nanoscale FinFET technology: — New-generation aging model and circuit reliability simulator," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2017, pp. 780–785.

[5] R. Huang et al., "Variability-and reliability-aware design for 16/14nm and beyond technology," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2017, pp. 1–4.

[6] J. Henkel et al., "Reliable on-chip systems in the nano-era: Lessons learnt and future trends," in *Proc. 50th ACM/EDAC/IEEE Des. Autom. Conf. (DAC)*, 2013, pp. 1–10.

[7] B. Tudor et al., "MOSRA: An efficient and versatile MOS aging modeling and reliability analysis solution for 45nm and below," in *Proc. 10th IEEE Int. Conf. Solid-State Integr. Circuit Technol.*, 2010, pp. 1645–1647.

[8] W. R. Davis, C. Shaw, and A. R. Hassan, "How to write a compact reliability model with the open model interface (OMI)," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, 2020, pp. 1–2.

[9] D. S. Huang et al., "Comprehensive device and product level reliability studies on advanced CMOS technologies featuring 7nm high-k metal gate FinFET transistors," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, 2018, pp. 1–5.

[10] "PrimeTime user guide." Synopsys. 2023. [Online]. Available: https://www.synopsys.com/cgi-bin/imp/pdfdla/pdfr1.cgi?file=primetime-wp.pdf

[11] J. B. Velamala, V. Ravi, and Y. Cao, "Failure diagnosis of asymmetric aging under NBTI," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2011, pp. 428–433.

[12] S. Kiamehr, F. Firouzi, M. Ebrahimi, and M. B. Tahoori, "Aging-aware standard cell library design," in *Proc. Des. Autom. Test Europe Conf. Exhibit. (DATE)*, 2014, pp. 1–4.

[13] Z. Zhang et al., "Aging-aware gate-level modeling for circuit reliability analysis," *IEEE Trans. Electron Devices*, vol. 68, no. 9, pp. 4201–4207, Sep. 2021.

[14] S. M. Ebrahimipour, B. Ghavami, H. Mousavi, M. Raji, Z. Fang, and L. Shannon, "Aadam: A fast, accurate, and versatile aging-aware cell library delay model using feed-forward neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2020, pp. 1–9.

[15] A. Vijayan, K. Chakrabarty, and M. B. Tahoori, "Machine learning-based aging analysis," in *Machine Learning in VLSI Computer-Aided Design*. Cham, Switzerland: Springer, 2019, pp. 265–289.

[16] N. Karimi and K. Huang, "Prognosis of NBTI aging using a machine learning scheme," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, 2016, pp. 7–10.

[17] W. Wang, Z. Wei, S. Yang, and Y. Cao, "An efficient method to identify critical gates under circuit aging," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2007, pp. 735–740.

[18] K.-C. Wu and D. Marculescu, "Aging-aware timing analysis and optimization considering path sensitization," in *Proc. Des. Autom. Test Europe*, 2011, pp. 1–6.

[19] Y. Lu, L. Shang, H. Zhou, H. Zhu, F. Yang, and X. Zeng, "Statistical reliability analysis under process variation and aging effects," in *Proc. 46th Annu. Des. Autom. Conf.*, 2009, pp. 514–519.

[20] "HSPICE user guide." Synopsys. 2023. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/ams-simulation/primesim-hspice.html

[21] F. Firouzi, F. Ye, K. Chakrabarty, and M. B. Tahoori, "Representative critical-path selection for aging-induced delay monitoring," in *Proc. IEEE Int. Test Conf. (ITC)*, 2013, pp. 1–10.

[22] A. F. Gomez and V. Champac, "Selection of critical paths for reliable frequency scaling under BTI-aging considering workload uncertainty and process variations effects," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 3, pp. 1–21, 2018.

[23] S. Salamin, G. Zervakis, O. Spantidi, I. Anagnostopoulos, J. Henkel, and H. Amrouch, "Reliability-aware quantization for anti-aging NPUs," in *Proc. Des. Autom. Test Europe Conf. Exhibit. (DATE)*, 2021, pp. 1460–1465.

[24] M. Ebrahimi, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Aging-aware logic synthesis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2013, pp. 61–68.

[25] "PrimeLib user guide." Synopsys. 2023. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/signoff/primelib.html

[26] I. Tsiokanos, L. Mukhanov, and G. Karakonstantis, "Low-power variation-aware cores based on dynamic data-dependent bitwidth truncation," in *Proc. Des. Autom. Test Europe Conf. Exhibit. (DATE)*, 2019, pp. 698–703.

[27] J. Constantin, L. Wang, G. Karakonstantis, A. Chattopadhyay, and A. Burg, "Exploiting dynamic timing margins in microprocessors for frequency-over-scaling with instruction-based clock adjustment," in *Proc. Des. Autom. Test Europe Conf. Exhibit. (DATE)*, 2015, pp. 381–386.

[28] J.-J. Liou, L.-C. Wang, A. Krstic, and K.-T. Cheng, "Experience in critical path selection for deep sub-micron delay test and timing validation," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2003, pp. 751–756.

[29] T. Liu, C.-C. Chen, and L. Milor, "Comprehensive reliability-aware statistical timing analysis using a unified gate-delay model for microprocessors," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 2, pp. 219–232, Apr.–Jun. 2018.

[30] K.-C. Wu and D. Marculescu, "Joint logic restructuring and pin reordering against NBTI-induced performance degradation," in *Proc. Des. Autom. Test Europe Conf. Exhibit.*, 2009, pp. 75–80.

[31] "Modelsim user guide." Mentor. 2023. [Online]. Available: https://eda.sw.siemens.com/en-US/ic/modelsim/

[32] "Design compiler user guide." Synopsys. 2023. [Online]. Available: https://www.synopsys.com/zh-cn/implementation-and-signoff/rtl-synthesis-test/design-compiler-graphical.html

[33] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980.*

[35] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM Conf. Manag. Data (SIGMOD)*, 2000, pp. 93–104.

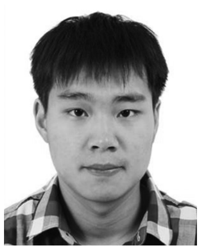
[36] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proc. ACM Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2007, pp. 824–833.

[37] B. Perozzi and L. Akoglu, "Scalable anomaly ranking of attributed neighborhoods," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2016, pp. 207–215.

[38] J. Li, H. Dani, X. Hu, and H. Liu, "Radar: Residual analysis for anomaly detection in attributed networks," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 2152–2158.

[39] Z. Peng, M. Luo, J. Li, H. Liu, and Q. Zheng, "ANOMALOUS: A joint modeling approach for anomaly detection on attributed networks," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 3513–3519.

[40] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2019, pp. 594–602.



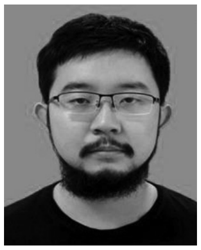
**Yuyang Ye** received the M.Sc. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2020. He is currently pursuing the Ph.D. degree with the National ASIC Research Center, Southeast University, Nanjing, China.

His current research interests include machine learning for EDA, timing analysis, and optimization.

Mr. Ye received the Best Student Paper Award from ICSICT 2022.

**Tinghuan Chen** (Member, IEEE) received the B.Eng. and M.Eng. degrees in electronics engineering from Southeast University, Nanjing, China, in 2014 and 2017, respectively, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2021.

He is currently an Assistant Professor with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. His research interests include machine learning for EDA and deep learning accelerators.



**Yifei Gao** received the B.Eng. degree from Southeast University, Nanjing, China, in 2021, where he is currently pursing the M.Eng. degree with the National ASIC Research Center.

His current research interests include timing analysis and optimization.

**Hao Yan** (Member, IEEE) received the B.S. degree from the Dalian University of Technology, Dalian, China, in 2011, and the M.S. and Ph.D. degrees from Southeast University, Nanjing, China, in 2014 and 2018, respectively.

He is currently an Associate Professor with National ASIC Research Center, Southeast University. His research focuses on design methodology for wide-voltage and high-efficiency design, including timing analysis and optimization.

**Bei Yu** (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received nine Best Paper Awards from DATE 2022, ICCAD 2021 and 2013, ASPDAC 2021 and 2012, ICTAI 2019, *Integration: The VLSI Journal* in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, and six ICCAD/ISPD contest awards. He has served as a TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is the Editor of IEEE TCCPS NEWSLETTER.

**Longxing Shi** (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Southeast University, Nanjing, China, in 1984, 1987, and 1992, respectively.

From 1992 to 2000, he was an Associate Professor with the School of Electronic Science and Engineering, Southeast University, where he has been a Professor and the Dean of the National ASIC Research Center since 2001. He has authored one book and over 130 articles. His current research interest includes ultra low-power IC design and design methodology.