Analog IC Aging-induced Degradation Estimation via Heterogeneous Graph Convolutional Networks

Tinghuan Chen
Chinese University of Hong Kong

Qi Sun
Chinese University of Hong Kong

Canhui Zhan
Hisilicon Technologies Co.

Changze Liu
Hisilicon Technologies Co.

Huatao Yu
Hisilicon Technologies Co.

Bei Yu
Chinese University of Hong Kong

Abstract

With continued scaling, transistor aging induced by Hot Carrier Injection and Bias Temperature Instability causes a gradual failure of nanometer-scale integrated circuits (ICs). In this paper, to characterize the multi-typed devices and connection ports, a heterogeneous directed multigraph is adopted to efficiently represent analog IC post-layout netlists. We investigate a heterogeneous graph convolutional network (H-GCN) to fast and accurately estimate aging-induced transistor degradation. In the proposed H-GCN, an embedding generation algorithm with a latent space mapping method is developed to aggregate information from the node itself and its multi-typed neighboring nodes through multi-typed edges. Since our proposed H-GCN is independent of dynamic stress conditions, it can replace static aging analysis. We conduct experiments on very advanced 5*nm* industrial designs. Compared to traditional machine learning and graph learning methods, our proposed H-GCN can achieve more accurate estimations of aging-induced transistor degradation. Compared to an industrial reliability tool, our proposed H-GCN can achieve 24.623× speedup on average.

1 Introduction

With continued scaling, the susceptibility of nanometer-scale transistors to aging-related wear-out phenomena has increased significantly in integrated circuits (ICs) [1]. As shown in Figures 1(a) and 1(b), two major aging-related wear-out mechanisms of semiconductor-based micro-electronic devices are bias temperature instability (BTI) and hot carrier injection (HCI) [2]. These aging effects cause transistor parameters, *e.g.*, threshold voltage, to shift from their nominal values over time, resulting in a gradual circuit failure [3–5]. Compared with digital ICs, the analog ICs are more susceptible to these transistor parameters.

In order to save development costs and provide the opportunity for interactive feedback during the design process, estimating aging-induced transistor degradation before committing the design to silicon is a key step in computer-aided design (CAD). Compared with pre-layout netlists, the aging analysis on post-layout netlists has more accurate judgment since post-layout netlists contain parasitic capacitances and resistances.

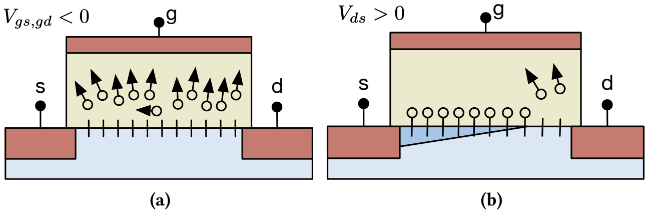


Figure 1: BTI and HCI mechanisms. (a) BTI: accumulated holes in silicon/oxide interface result in breaking of Si-H bonds; (b) HCI: due to high electric field in drain side, hot carriers cause breaking of Si-H bond and traps oxide bulk.

Modeling aging-induced transistor degradation has been studied in many works of literature. Tu *et al.* adopted a voltage-controlled current source model to identify reliability problems such as the HCI issue in the design stage [6]. A repetitive simulation scheme was applied to ensure an accurate prediction of the BTI- and HCI-induced circuit-level degradation process in Hot-Carrier Reliability Simulation [4]. In [3], a Reaction-Diffusion model was proposed to determine delay degradation caused by Negative BTI. Other analytical models were well surveyed and concluded in [7]. Moreover, a ring-oscillator-based sensor was used to calibrate aging-induced circuit performance degradation [1].

However, there are also several limitations and drawbacks of these analytical models. Firstly, a correct judgment on the reliability of the circuits heavily relies on the appropriate stress conditions for aging analysis [8]. It is hard to find the appropriate stress conditions that lead to the actual case on the aging-prone transistors. While the static aging analysis causes inaccurate judgment on the aging-prone transistors since the dynamic stress conditions are completely ignored. Secondly, it is time-consuming to achieve accurate detections when these models are implemented within the simulators. It is difficult to find a compromise between computational complexity and model accuracy for the model implementation.

Convolutional neural networks (CNNs) have achieved great success in CAD such as the design for manufacturability [9]. Intuitively, an analog IC netlist can be well represented as a graph. However, the graph is an irregular grid-based data, which is not as straightforward as the convolution and pooling in traditional CNNs. Graph convolutional networks (GCNs) were proposed to perform machine learning tasks on these irregular grid-based data [10, 11]. Recently, GCNs were adopted to predict observation point candidates on design for

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431546

testability [12]. However, the analog IC netlists have heterogeneity since typical analog ICs contain multi-typed basic devices and multi-typed connection ports. Therefore, traditional GCN cannot be directly used on analog IC netlists.

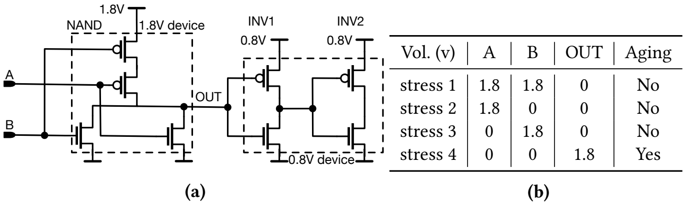
In this paper, considering multi-typed connection ports and devices with different design parameters, a heterogeneous directed multigraph is adopted to efficiently represent a post-layout netlist. Then we propose a heterogeneous GCN (H-GCN) to fast and accurately estimate aging-induced transistor degradation. In our proposed H-GCN, an embedding generation algorithm with a latent space mapping method is developed to aggregate information from the node itself and its multi-typed neighboring nodes through multi-typed edges. Since our proposed H-GCN is independent of dynamic stress conditions, it can replace static aging analysis. A neighborhood sampling method is used to ease large scale circuit training and reduce GPU memory overhead. We conduct experiments on advanced 5*nm* industrial designs to show the proposed H-GCN can achieve more accurate estimations of aging-induced transistor degradation.

2 Problem Formulation

Before committing the design to silicon, aging-induced transistor degradation is required to accurately estimate in the post-layout simulation to judge circuit reliability. However, the actual degree of the aging-induced transistor degradation is hard to be estimated by using the traditional aging analysis. The traditional static aging analysis causes inaccurate judgment on the aging-prone transistors since the dynamic stress conditions are completely ignored. While the traditional dynamic aging analysis heavily relies on the dynamic stress conditions such as clock speeds and waveform swings. As an example in Figure 2, there are four combinations of stresses. However, there is only one stress resulting in the device aging. Furthermore, if there are massive combinations for stresses in circuits, it is expensive and time-consuming to obtain the actual degree of the aging-induced transistor degradation. `dvtlin (HCI+BTI)` is used to assess the degree of aging-induced transistor degradation in the industry [5]. `dvtlin (HCI+BTI)` is defined as follows:

Definition 1 (`dvtlin (HCI+BTI)`). `dvtlin (HCI+BTI)` is defined as the shifting value of threshold voltage of the transistor from fresh to 10 years due to HCI and BTI.

The larger `dvtlin (HCI+BTI)` is, the worse aging-induced transistor degradation is, vice versa. For convenience, in this paper, we shorten `dvtlin (HCI+BTI)` as `dvtlin`.



Vol. (v)	A	B	OUT	Aging
stress 1	1.8	1.8	0	No
stress 2	1.8	0	0	No
stress 3	0	1.8	0	No
stress 4	0	0	1.8	Yes

Figure 2: Aging effects and stress conditions in the circuit.

Based on the above description, we define our problem formulation as follows.

Problem 1 (Estimating `dvtlin` in Analog ICs). Given some analog IC post-layout netlists and a list containing all transistors with `dvtlin` obtained by the dynamic aging analysis as the training set, our task is training a model on the training set to fast and accurately estimate the `dvtlin` of transistors on the testing set while minimizing the estimation error.

3 Heterogeneous Graph Representation

To perform an embedding generation algorithm in a GCN-based framework, we treat each device, direct current (DC) voltage source or ground as a node and represent the analog IC post-layout netlists as a graph. Note that the dynamic stresses are not considered so that our model can be independent of them. A Naïve method is to construct a homogeneous undirected multigraph to represent the circuit netlist. In this homogeneous undirected multigraph, each undirected edge represents one path from one device to another device. It allows multiple edges between any two nodes since there may be multiple paths between two devices.

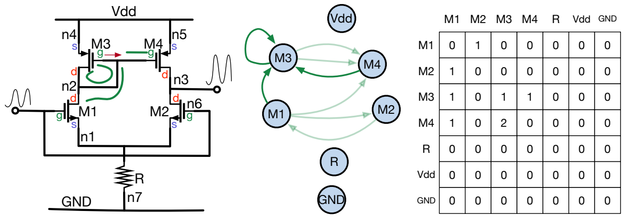
The typical analog IC netlists have heterogeneity since they contain multi-typed basic devices and multi-typed connection ports. An inevitable problem of the homogeneous representation method is that it fails to characterize the diversities among ports, devices, connections, and relative sequential relationships. We set the type of the edges as the type of the ports to which it connects. In order to express these diversities, we will propose a heterogeneous directed multigraph representation, which is defined as follows.

Definition 2 (Heterogeneous directed multigraph). A heterogeneous directed multigraph is defined as a graph $\mathcal{HMG}_d(\mathcal{V}_{hmg}, \mathcal{E}_{hmg}, \mathcal{O}_{\mathcal{V}_{hmg}}, \mathcal{R}_{\mathcal{E}_{hmg}}, \varphi_{hmg})$, where $\mathcal{V}_{hmg}$ is the set of nodes, $\mathcal{E}_{hmg}$ is the multiset of edges. $\mathcal{O}_{\mathcal{V}_{hmg}}$ and $\mathcal{R}_{\mathcal{E}_{hmg}}$ represent the sets of node types and edge types, respectively. φ_{hmg} is adopted to assign each node in $\mathcal{V}_{hmg}$ to a node type, *i.e.*, $\varphi_{hmg}(v_i) \in \mathcal{O}_{\mathcal{V}_{hmg}}$ for $\forall v_i \in \mathcal{V}_{hmg}$. r indicates the edge type, such that $r \in \mathcal{R}_{\mathcal{E}_{hmg}}$. Each instance in $\mathcal{E}_{hmg}$ is $((v_i, v_j), r)$ and the ordered pair (v_i, v_j) satisfies $v_i, v_j \in \mathcal{V}_{hmg}$.

We use an example to show this heterogeneous directed multigraph representation. As shown in Figure 3(a), if we stand at the gate of the transistor `M3` and lookout, we will see transistors `M1`, `M3` and `M4`. Thus there are three directed edges with gate connections from `M1`, `M3` and `M4` to the transistor `M3` as shown in Figure 3(b). This method is inspired by circuit analysis, where the input impedance is obtained in the same fashion. In the same manner, we can obtain the directed multigraph corresponding to the gate connections as shown in Figure 3(b). Furthermore, the circuit netlist in Figure 3(a) is finally transformed into the multigraph in Figure 4.

In particular, considering that the electrical characteristics of direct current voltage sources and grounds are not influenced by other devices, we set them to be predecessors.

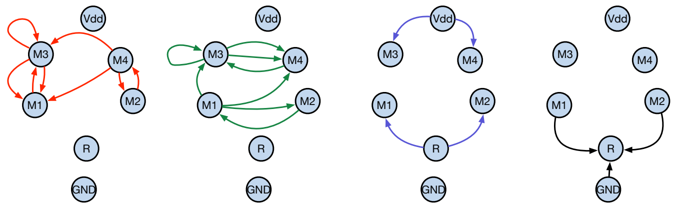
We use one adjacency matrix to represent one type of edges in the heterogeneous multigraph. The adjacency matrix of the type r connection is defined as $\boldsymbol{A}_r$, where each element $A_r(i, j)$ is the number of the instance $((v_i, v_j), r)$ in the multiset $\mathcal{E}_{dmg}$. As an example, the adjacency matrix of the gate connection in the heterogeneous directed multigraph in Figure 3(b) is shown in Figure 3(c).



	M1	M2	M3	M4	R	Vdd	GND
M1	0	1	0	0	0	0	0
M2	1	0	0	0	0	0	0
M3	1	0	1	1	0	0	0
M4	1	0	2	0	0	0	0
R	0	0	0	0	0	0	0
Vdd	0	0	0	0	0	0	0
GND	0	0	0	0	0	0	0

(a) **(b)** **(c)**

Figure 3: The heterogeneous directed multigraph representation. (a) The differential amplifier; (b) The gate connection; (c) The adjacency matrix.



(a) **(b)** **(c)** **(d)**

Figure 4: A heterogeneous directed multigraph with multi-typed edges. (a) Drain; (b) Gate; (c) Source; (d) Others.

4 Heterogeneous GCN

4.1 Notations

As discussed in Section 3, given an analog IC netlist, we can transform it to be a heterogeneous directed multigraph $\mathcal{HMG}_d(\mathcal{V}_{hmg}, \mathcal{E}_{hmg}, \mathcal{O}_{\mathcal{V}_{hmg}}, \mathcal{R}_{\mathcal{E}_{hmg}}, \varphi_{hmg})$. Then we can obtain $|\mathcal{R}_{\mathcal{E}_{hmg}}|$ adjacency matrices, *i.e.*, $\boldsymbol{A}_r$ for $\forall r \in \mathcal{R}_{\mathcal{E}_{hmg}}$. Besides, the device $v_i \in \mathcal{V}_{hmg}$ has $h_{\varphi_{hmg}(v_i)}$ design parameters for $\varphi_{hmg}(v_i) \in \mathcal{O}_{\mathcal{V}_{hmg}}$. Note that the number of design parameters relies on the type of device. We use these design parameters as the node features, *i.e.*, $\boldsymbol{x}_{v_i} \in \mathbb{R}^{1 \times h_{\varphi_{hmg}(v_i)}}$ and $\varphi_{hmg}(v_i) \in \mathcal{O}_{\mathcal{V}_{hmg}}$ for $\forall v_i \in \mathcal{V}_{hmg}$.

4.2 Unified Latent Space Mapping

A typical analog IC netlist contains multiple types of devices, which have different design parameters. We regard each device as a node and use its design parameters as the node features vector. To perform embedding generation algorithm and aggregate information from the node itself and its multi-typed neighboring nodes, a straightforward method is to concatenate all of these design parameters as a long feature vector with one-hot encoding. However, it will miss some structural information among multi-typed nodes as well as unstructured content associated with each node [13]. In this paper, a latent space mapping method is used to transform the features vectors of all multi-typed nodes into a unified latent space.

We propose to use node-type-related mapping matrices to map the original feature vectors. For a node $v_i \in \mathcal{V}_{hmg}$ whose node type is $t = \varphi_{hmg}(v_i) \in \mathcal{O}_{\mathcal{V}_{hmg}}$ and feature vector is $\boldsymbol{x}_{v_i} \in \mathbb{R}^{1 \times h_t}$, we define a node-typed-related matrix $\boldsymbol{U}_t \in \mathbb{R}^{h_t \times \lambda}$ to map the feature vector with length h_t into a unified λ-dimension latent space, *i.e.*, $\boldsymbol{f}_{v_i} = \boldsymbol{x}_{v_i} \cdot \boldsymbol{U}_t \in \mathbb{R}^{1 \times \lambda}$. In order to ease the model training on GPUs, we extend it to be a matrix-matrix multiplication as follows:

$$\boldsymbol{F} = \sum_{t \in \mathcal{O}_{\mathcal{V}_{hmg}}} \boldsymbol{X}_t \cdot \boldsymbol{U}_t \in \mathbb{R}^{|\mathcal{V}_{hmg}| \times \lambda}, \tag{1}$$

where $\boldsymbol{X}_t \in \mathbb{R}^{|\mathcal{V}_{hmg}| \times h_t}$ is a feature matrix stacking the feature vectors of all type-t nodes. In particular, a node feature vector is $\boldsymbol{0} \in \mathbb{R}^{1 \times h_t}$ if the type of the node is not t. $\boldsymbol{F}$ is the feature matrix, where the feature vectors of all nodes are in a unified latent space.

As an example, there are three types of nodes (devices) in Figure 3(a), *i.e.*, $\mathcal{O}_{\mathcal{V}_{hmg}} = \{vs, trans, res\}$. Thus there are three feature matrices: $\boldsymbol{X}_{vs} = [\boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{x}_{Vdd}^\top, \boldsymbol{x}_{GND}^\top]^\top$, $\boldsymbol{X}_{trans} = [\boldsymbol{x}_{M1}^\top, \boldsymbol{x}_{M2}^\top, \boldsymbol{x}_{M3}^\top, \boldsymbol{x}_{M4}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top]^\top$ and $\boldsymbol{X}_{res} = [\boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{x}_{R}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top]^\top$.

It is noted that the concatenated features representation is a special case of latent space representation. Compared with the concatenated features representation, our latent space representation is general enough and can effectively exploit and encode features.

4.3 Embedding Generation

As mentioned above, the analog IC netlists are represented as heterogeneous directed multigraphs. After applying the proposed latent space mapping method, the features of nodes share the same representation forms. An H-GCN model is proposed as the skeleton of the estimation model, and the inputs are the unified latent feature representations of all nodes and adjacency matrices corresponding to the heterogeneous multigraph.

As discussed in Section 3, we use one adjacency matrix to represent one type of edges in the heterogeneous directed multigraph. In order to distinguish among different connection ports, inspired by [12], we assign a learnable model coefficient to each adjacency matrix. Then the overall connection of the heterogeneous directed multigraph can be expressed as the summation of the adjacency matrices of all types of edges with these model coefficients. For example, as shown in Figure 4, there are four adjacency matrices $\mathbf{A}_g$, $\mathbf{A}_s$, $\mathbf{A}_d$ and $\mathbf{A}_{otr}$ in the heterogeneous directed multigraph. We assign four model coefficients w_g, w_s, w_d and w_{otr} to distinguish them. Therefore, the connections of the heterogeneous directed multigraph can be expressed as $w_g\mathbf{A}_g + w_s\mathbf{A}_s + w_d\mathbf{A}_d + w_{otr}\mathbf{A}_{otr}$.

In our proposed H-GCN, we develop an novel embedding generation algorithm to aggregate information of the node itself and its multi-typed neighboring nodes through multi-typed edges as follows:

$$\boldsymbol{F}_{\mathcal{N}}^{(l-1)} = \left(\sum_{r \in \mathcal{R}_{\mathcal{E}_{hmg}}} w_r \boldsymbol{A}_r \right) \cdot \boldsymbol{F}^{(l-1)}, \tag{2}$$

$$\boldsymbol{F}^{(l)} = \sigma \left(\mathrm{CONCAT} \left(\boldsymbol{F}_{\mathcal{N}}^{(l-1)}, \boldsymbol{F}^{(l-1)} \right) \cdot \boldsymbol{W}^{(l)} \right), \tag{3}$$

where $\boldsymbol{W}^{(l)}$ is learnable model coefficients in the l-th layer. $\boldsymbol{F}_{\mathcal{N}}^{(l)}$ is the feature representation of neighboring nodes in the l-th layer. $\boldsymbol{F}^{(l)}$ is the feature representation of all nodes in the l-th layer. w_r is a model coefficient for the type-r connection. CONCAT$(\cdot)$ denotes the concatenation operation. Our proposed embedding generation algorithm (*i.e.*, forward propagation) is summarized in Algorithm 1.

Algorithm 1 Embedding Generation of H-GCN

```
Input: $|\mathcal{R}_{\mathcal{E}_{hmg}}|$ adjacency matrices $\boldsymbol{A}_r$ and connection-type-related model coefficients $w_r$ with $\forall r \in \mathcal{R}_{\mathcal{E}_{hmg}}$; feature matrices of all nodes corresponding to each type of nodes $\boldsymbol{X}_t$ and node-type-related latent space mapping matrices $\boldsymbol{U}_t$ with $\forall t \in \mathcal{O}_{\mathcal{V}_{hmg}}$; Search depth $D$; non-linear activation function $\sigma(\cdot)$; Model coefficients matrices $\boldsymbol{W}^{(l)}$.
$\boldsymbol{F}^{(0)} \leftarrow \sum_{t \in \mathcal{O}_{\mathcal{V}_{hmg}}} \boldsymbol{X}_t \boldsymbol{U}_t$;
for $l = 0$ to $D - 1$ do
    $\boldsymbol{F}_{\mathcal{N}}^{(l)} \leftarrow \left(\sum_{r \in \mathcal{R}_{\mathcal{E}_{hmg}}} w_r \boldsymbol{A}_r\right) \cdot \boldsymbol{F}^{(l)}$;
    $\boldsymbol{F}^{(l+1)} \leftarrow \sigma\left(\text{CONCAT}\left(\boldsymbol{F}_{\mathcal{N}}^{(l)}, \boldsymbol{F}^{(l)}\right) \cdot \boldsymbol{W}^{(l)}\right)$;
end for
return Embedding feature matrix $\boldsymbol{F}^{(D)}$.
```

Algorithm 1 describes the embedding generation process, where multiple adjacency matrices $\boldsymbol{A}_r$, connection-type-related model coefficients w_r, feature matrices of all nodes corresponding to each type of nodes $\boldsymbol{X}_t$, node-type-related latent space mapping matrices $\boldsymbol{U}_t$ are provided as inputs. According to Equation (1), the latent space mapping matrices $\boldsymbol{U}_t$ are used to map features of all nodes $\boldsymbol{X}_t$ into a unified feature representation $\boldsymbol{F}^{(0)}$. Let l denotes the current step in the loop (the depth of the search) and $\boldsymbol{F}^{(l)}$ denotes the feature representation of all nodes at this step. Then each step in the loop of Algorithm 1 proceeds as follows: Firstly, all nodes aggregate the feature representations of the nodes in their immediate neighborhood into a matrix $\boldsymbol{F}_{\mathcal{N}}^{(l)}$. Then the aggregated neighboring feature matrix $\boldsymbol{F}_{\mathcal{N}}^{(l)}$ is concatenated with the node's current representation $\boldsymbol{F}^{(l)}$. And this concatenated feature matrix is fed through a fully connected (FC) layer with nonlinear activation function $\sigma(\cdot)$ which makes the model more robust. The outputs are used at the next step of the algorithm.

Note that only transistors are prone to have aging degradations in analog IC netlist while features of all devices need to be fed into our proposed H-GCN to perform the embedding generation algorithm. Thus a mask matrix with size *#transistors* × *#devices* is used to extract the estimated `dvtlin` of transistors in the last embedding layer. Then several traditional FC layers are adopted to construct the relationships between embedding features and `dvtlin`. Finally, the H-GCN generates an output that represents the estimated `dvtlin` of transistors. Mean Square Error (MSE) function is used as the loss function to compute the errors between the ground-truth and our estimated `dvtlin`. All model coefficients, including w_r, $\boldsymbol{U}_t$ and $\boldsymbol{W}^{(l)}$ in Algorithm 1, are obtained by the back propagation and the gradient-based method in end-to-end fashion in the training stage.

4.4 Large Scale Graph Training via Neighborhood Sampling

With the fast development of semiconductors, massive devices and connections are integrated into an analog IC, which leads to great challenges in memory overhead to the model training because of the iterative embedding generations among neighboring nodes on large graphs.

To achieve enough scalability on large graphs, a probability-based neighborhood sampling algorithm is proposed, inspired by [10, 14]. In each back-propagation process, whether one neighboring node is computed in the gradients is determined by the probability proportional to the degree of this neighboring node. Note that although the original analog IC netlist has been represented as a heterogeneous directed multigraph with $|\mathcal{R}_{\mathcal{E}_{hmg}}|$ adjacency matrices $\boldsymbol{A}_r$ ($r \in \mathcal{R}_{\mathcal{E}_{hmg}}$), the sampling probability is compuated according to the overall adjacency matrix $\boldsymbol{A} \triangleq \sum_{r \in \mathcal{R}_{\mathcal{E}_{hmg}}} \boldsymbol{A}_r$. The sampling probability is given by Equation (4).

$$\mathcal{P}(v_i) = \frac{|\boldsymbol{A}(i,:)| + |\boldsymbol{A}(:,i)| - \boldsymbol{A}(i,i)}{|\boldsymbol{A}|}, \tag{4}$$

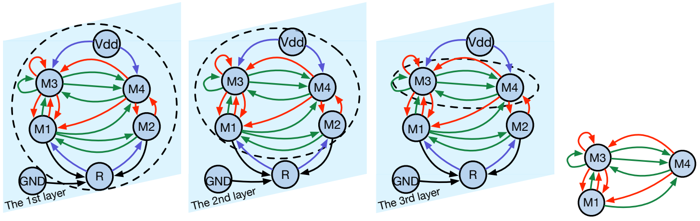
where $|\boldsymbol{A}|$ is summation of all of the elements in matrix $\boldsymbol{A}$, *i.e.*, the summation of degrees of all of the nodes, and $|\boldsymbol{A}(i,:)|$ ($|\boldsymbol{A}(:,i)|$) is the summation of the i-th row (column) in $\boldsymbol{A}$. $\boldsymbol{A}(i,i)$ is the number of self-loops of the node v_i.

As shown in Algorithm 2, three are D aggregation layers in the H-GCN model. A threshold T determines whether the neighborhood set should be sampled to reduce the training overhead. Starting from the input node set $\mathcal{V}_s$, we iteratively traverse the graph and add all of the nodes we have visited in the k-hop neighborhood into $\mathcal{V}^{(k)}$. k-hop neighborhood is the set of nodes at the distance less than or equal to k from the node itself. All of the nodes we have visited in the D-hop neighborhood are finally stored in a node set $\mathcal{V}^{(D)}$. Except the starting set $\mathcal{V}_s$, all of the other nodes in $\mathcal{V}^{(D)}$ are sampled according to the probability defined in Equation (4) into the final node set $\mathcal{V}_b$. $\mathcal{V}_b$ is the sampled node set of $\mathcal{V}_s$. While computing the gradients of nodes in $\mathcal{V}_s$, only the neighbor nodes in $\mathcal{V}_b$ are considered. Our proposed neighborhood sampling algorithm can be naturally used in the inference stage.

Algorithm 2 The Neighborhood Sampling Algorithm

```
Input: The input node set $\mathcal{V}_s$, neighborhood threshold $T$.
$\mathcal{V}^{(0)} \leftarrow \mathcal{V}_s$;
for $k = 1$ to $D$ do
    $\mathcal{V}^{(k)} \leftarrow \mathcal{V}^{(k-1)}$;
    for all $\mu \in \mathcal{V}^{(k-1)}$ do
        $\mathcal{V}^{(k)} \leftarrow \mathcal{V}^{(k)} \cup \mathcal{N}(\mu)$;
    end for
end for
if $|\mathcal{V}^{(D)}| > T$ then
    for all $v_i \in \mathcal{V}^{(D)} \backslash \mathcal{V}_s$ do
        Calculate $\mathcal{P}(v_i)$ according to Equation (4);
    end for
    Sample $T - |\mathcal{V}^{(D)}|$ nodes according to the probability $\mathcal{P}$ to the set $\mathcal{V}_b$;
else
    $\mathcal{V}_b \leftarrow \mathcal{V}^{(D)}$;
end if
return The sampled node set $\mathcal{V}_b$;
```

Figure 5(a) is an example to illustrate the sampling algorithm. There are three aggregation layers and the sampling threshold T is set to 3. The initial input node set is $\mathcal{V}_s = \{$`M3`, `M4`$\}$. The 3-hop neighborhood node set $\mathcal{V}^{(3)}$ is $\{$`M1`, `M2`, `M3`, `M4`, `Vdd`, `R`, `GND`$\}$. Except for `M3` and `M4` in $\mathcal{V}_s$, we only need to sample $T - |\mathcal{V}_s| = 1$ more node from $\mathcal{V}^{(D)} \backslash \mathcal{V}_s$. According to Equation (4), `M1` has the largest



(a) (b)

Figure 5: (a) Neighborhood expansions; (b) The sampled set $\mathcal{V}_b$.

probability to be sampled. If M1 is sampled, the output node set is $\mathcal{V}_b = \{\texttt{M1}, \texttt{M3}, \texttt{M4}\}$ as shown in Figure 5(b). While computing the gradients of M3 and M4, only $\mathcal{V}_b$ is computed.

5 Experimental Results

The experiments are illustrated on five different industrial phase-locked loop designs implemented in very advanced $5nm$ technology node. These designs are represented with post-layout netlists. Because post-layout netlists have parasitic capacitances and resistances information, compared to pre-layout netlists, the aging analysis on post-layout netlists can achieve more accurate judgements. The post-layout netlist files consist of Spectre and SPICE formats, as inputs in our models. We run an industrial dynamic aging design for reliability (DFR) tool to obtain the dvtlin value of each transistor. Note that in each design, the stress conditions are given by very sophisticated designers to obtain dvtlin as golden-truth value. Compared the traditional static aging analysis, our model has more accurate estimations since it is trained with degradation obtained from the industrial aging DFR tool with the appropriate stress. To evaluate the estimation performance, each time, we use one of five designs for testing and all others four designs for training.

The statistics of the five industrial phase-locked loop designs are shown in Table 1. Each design is parsed and flattened. Then all alternating current (AC) voltage sources (dynamic conditions), such as behavioral signal (Verilog-A description), sine wave, pulse and piecewise linear (pwl), are ignored while all DC voltage sources are considered so that our model can be independent of the dynamic stress conditions. Except for AC voltage sources, there are 32 types of basic devices in all designs. All of these devices are divided into 6 categories as listed in Table 2. Their feature vectors have different lengths, *i.e.*, the #Feature. Correspondingly, 6 matrices are adopted to map feature vectors into a unified latent space.

According to the domain knowledge, the four ports of transistors and two ports of diodes play an important role in circuit analysis. Consequently, in the heterogeneous graph representation, all of the edges connecting to these six types of ports are emphasized by categorizing them into six types of edges. All of the edges connecting to other ports are uniformly treated as an individual type of edges. In total, seven adjacency matrices are used to specify various connection ports, *i.e.*, corresponding to gate, drain, source, substrate, anode, cathode and others. One-hot representation method is used to encode the type vector.

Four baselines are implemented in the experiments, as shown in Table 3. All of these four baseline models use concatenated features. Two are traditional machine learning methods: Gaussian Process Regression (GPR), and MLP. The traditional GCN [10] which treats the analog IC netlist as homogeneous undirected multigraphs mentioned in Section 3 is compared with our proposed H-GCN to prove the effectiveness of our heterogeneous directed multigraph representation and heterogeneous embedding generation. As an ablation study, our H-GCN-concat uses the proposed heterogeneous directed multigraph representation and concatenation feature representation to verify the effectiveness of our unified latent space mapping algorithm. In other words, the difference between H-GCN and H-GCN-concat is that H-GCN uses the proposed unified latent space mapping algorithm while H-GCN-concat adopts concatenation feature representation.

In our H-GCN model, ReLU function is used as the activation function. To improve numerical stability, we normalize all feature vectors and adjacency matrices. We use MSE between the ground-truth and our estimated dvtlin as loss function with weight decay hyperparameter 10^{-7} and stochastic gradient descent for optimization. To make a trade-off between GPU memory efficiency and accuracy, we use the proposed embedding generation with depth $D = 2$ shown in Algorithm 1 and 3 FC layers in our proposed H-GCN. Thanks to our proposed neighborhood sampling, GCN, H-GCN-concat and H-GCN have enough scalability on the large graph, whose node number is up to $185K$ in Table 1. And we set the node sampling size $T = 1000$ in Algorithm 2. Each output feature size is 512. Batch size is 32 and the number of the training epoch is 500. These configurations and parameters are determined by grid search. MLP has five FC layers. The traditional GCN and H-GCN-concat are both composed of two embedding layers and 3 FC layers. In conclusion, MLP, GCN and our H-GCN-concat have the same configurations with H-GCN.

The proposed graph representation is implemented with Python and Networkx library. GCN models are implemented with TensorFlow, and are trained and tested on a Linux machine with 18 cores and NVIDIA Tesla V100 GPU with 32GB memory. The baseline machine learning methods are implemented with Python and scikit-learn.

Mean absolute error (MAE) defined in Equation (5) is used as a metric to evaluate the absolute accuracy on the testing set.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{5}$$

where i indicates the i-th transistor. y_i is dvtlin, as the golden-truth value, obtained by the industrial dynamic aging DFR tool with the appropriate stress given by very sophisticated designers. $\hat{y}_i$ is the estimated dvtlin by machine learning or graph learning methods. n is the number of transistors on the testing set. In addition, we use r^2 score defined in Equation (6) as a metric to evaluate the relative accuracy on the testing set.

$$r^2\text{score} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{6}$$

where $\bar{y}$ denotes the mean of the golden-truth values y_i on the testing set. r^2 score indicates how the regression prediction perfectly fits the data. The higher r^2 means the better relative accuracy.

The MAE results are shown in Figure 6(a). Our proposed H-GCN estimates dvtlin with the lowest $1.701mV$ error on average and outperforms all of these four baselines by a lot. Compared with traditional GCN, our H-GCN-concat is much better which shows that

Table 1: Statistics of Designs

Design	#trans.	#device	#net
1	4348	99009	18155
2	4382	99696	18299
3	3999	179758	31303
4	3998	185480	33819
5	523	31279	6002

Table 2: Types of Devices

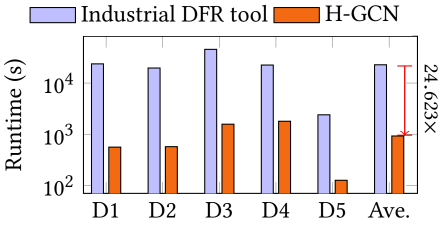
Type	#Design parameters
MOS	51
MOS spice	75
DIO/ESD	8
Cap	12
R	6
VSource	1

Table 3: Baseline Algorithms

Method	Feature		Graph representation	
	Concat.	Latent	Homo.	Heter.
GPR	✓		/	/
MLP	✓		/	/
GCN	✓		✓	
Our H-GCN-concat	✓			✓
Our H-GCN		✓		✓

Design	GPR		MLP		GCN [10]		Our H-GCN-concat		Our H-GCN	
	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score
1	3.394	0.366	3.037	0.370	1.416	0.618	1.307	0.775	**1.060**	**0.807**
2	3.481	0.448	3.283	0.540	1.395	0.601	1.355	0.763	**1.070**	**0.784**
3	5.041	0.257	4.736	0.415	4.404	0.558	2.771	0.760	**2.462**	**0.791**
4	4.985	0.245	4.850	0.401	4.408	0.548	2.811	0.749	**2.487**	**0.753**
5	4.641	0.393	4.230	0.595	2.344	0.792	1.861	0.818	**1.427**	**0.851**
Ave.	4.308	0.342	4.027	0.464	2.793	0.627	2.021	0.773	**1.701**	**0.797**

(a)



(b)

Figure 6: (a) MAE (mV) and r^2 Score Comparisons; (b) Runtime comparison with the industrial DFR tool.

our heterogeneous directed multigraph representation algorithm improves the performance significantly. Our H-GCN is also better than our H-GCN-concat, which further verify the effectiveness of our unified latent space mapping algorithm.

As shown in Figure 6(a), compared with other methods, our H-GCN achieves the highest r^2 scores on all of these five designs. Therefore, according to Figure 6(a), our proposed H-GCN can achieve the best accuracy.

We also compare the runtime of our proposed H-GCN with the industrial dynamic aging DFR tool with an appropriate stress as shown in Figure 6(b). For fairness, the runtime of our proposed H-GCN is composed of the graph representation and H-GCN inference shown in Algorithm 1. As shown in Figure 6(b), compared with the industrial dynamic aging DFR tool, our H-GCN achieves the faster runtime on all of these five designs. On average, our proposed H-GCN can achieve 24.623× speedup. In practice, as mentioned in Section 2, the worst case is hard to find by the traditional dynamic aging simulator since all cases of stress need be considered. So compared with the industrial dynamic aging DFR tool, our proposed H-GCN can achieve further speedup.

6 Conclusion

In this paper, a heterogeneous directed multigraph is adopted to efficiently represent analog IC post-layout netlists. Then we propose an H-GCN to fast and accurately estimate aging-induced transistor degradation. We develop an embedding generation algorithm with a latent space mapping to aggregate information from the node itself and its multi-typed neighboring nodes through multi-typed edges. A neighborhood sampling method is used to ease the model training on large scale graphs and reduce GPU memory overhead. Since our proposed H-GCN is independent of dynamic stress conditions, it can replace static aging analysis. We conduct experiments on very advanced 5*nm* industrial designs. Compared to traditional machine learning and graph learning methods, the proposed H-GCN can achieve more accurate estimations of aging-induced transistor degradation. Compared to an industrial dynamic aging DFR tool, our proposed H-GCN can achieve 24.623× speedup on average.

Acknowledgment

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14209420) and HiSilicon. The authors would like to thank Dr. Chao Yang from HiSilicon for helpful discussions.

References

[1] D. Sengupta and S. S. Sapatnekar, "Estimating circuit aging due to BTI and HCI using ring-oscillator-based sensors," *IEEE TCAD*, vol. 36, no. 10, pp. 1688–1701, 2017.

[2] Z. Yu, Z. Sun, R. Wang, J. Zhang, and R. Huang, "Hot carrier degradation-induced dynamic variability in FinFETs: Experiments and modeling," *IEEE TED*, vol. 67, no. 4, pp. 1517–1522, 2020.

[3] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An analytical model for negative bias temperature instability," in *Proc. ICCAD*, 2006, pp. 493–496.

[4] M. Karam, W. Fikry, H. Haddara, and H. Ragai, "Implementation of hot-carrier reliability simulation in Eldo," in *Proc. ISCAS*, vol. 5, 2001, pp. 515–518.

[5] *New Generation Reliability Model*, 2016, http://www.mos-ak.org/berkeley_2016/publications/T11_Xie_MOS-AK_Berkeley_2016.pdf.

[6] R. H. Tu, E. Rosenbaum, W. Y. Chan, C. C. Li, E. Minami, K. Quader, P. K. Ko, and C. Hu, "Berkeley reliability tools-BERT," *IEEE TCAD*, vol. 12, no. 10, pp. 1524–1534, 1993.

[7] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, "Electronic circuit reliability modeling," *Microelectronics Reliability*, vol. 46, no. 12, pp. 1957–1979, 2006.

[8] C. Schlünder, K. Waschneck, P. Rotter, S. Lachenmann, H. Reisinger, F. Ungar, and G. Georgakos, "From device aging physics to automated circuit reliability sign off," in *Proc. IRPS*, 2019, pp. 1–12.

[9] Y. Jiang, F. Yang, H. Zhu, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network," in *Proc. DAC*, 2019, p. 147.

[10] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1024–1034.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2016.

[12] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu, "High performance graph convolutional networks with applications in testability analysis," in *Proc. DAC*, 2019, p. 18.

[13] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proc. KDD*, 2015, pp. 119–128.

[14] J. Chen, T. Ma, and C. Xiao, "FastGCN: fast learning with graph convolutional networks via importance sampling," *Proc. ICLR*, 2018.