Linear Time Electromigration Analysis Based on Physics-Informed Sparse Regression

Liang Chen, *Member, IEEE*, Wentian Jin, *Student Member, IEEE*, Mohammadamir Kavousi, *Graduate Student Member, IEEE*, Subed Lamichhane, *Student Member, IEEE*, and Sheldon X.-D. Tan, *Senior Member, IEEE*

***Abstract*—In this work, we propose a novel physics-informed sparse regression (PISR) framework to solve stress evolution (described by Korhonen's equations) in general multisegment wires using an unsupervised learning scheme. Unlike the existing physics-informed neural network (PINN) framework, the PISR method trains the trainable weights through the Moore–Penrose generalized inverse algorithm used in extreme learning machine (ELM), which is extremely faster than the backpropagation algorithm. To improve the accuracy of PISR for complex multisegment interconnects, we employ domain decomposition schemes in both space and time. For each subdomain, we use different trainable weights but the same shared neural network to represent each subsolution, which leads to more efficient memory usage. Furthermore, we propose to use sparse matrix techniques to accelerate the training speed of the PISR method and prove that the resulting PISR has linear time complexity for analyzing tree-structured interconnects. Finally, we divide the time into many time intervals and apply an autoregressive model to simulate each time interval in sequence to further improve scalability and reduce memory cost so that the PISR method can perform EM analysis for large-scale multisegment interconnects. Experimental results on different kinds of interconnect structures show that the proposed PISR method has the same accuracy level as the numerical methods. The results on N_T T-junctions interconnect trees show that the proposed PISR method indeed demonstrates true *linear* time complexity. Furthermore, PISR can deliver 8.9×, 20.6×, and 1284× speedups over the recently proposed semi-analytic method (ASOV), finite difference method accelerated with model order reduction (FDM-MOR), FDM for the interconnect with $N_T = 5000$, respectively. Furthermore, we show that PISR also achieves an 818× speedup in training over the plain PINN method based on the traditional backpropagation algorithm.**

***Index Terms*—Autoregressive model, electromigration (EM), physics-informed sparse regression (PISR), space-time domain decomposition method.**

I. Introduction

ELECTROMIGRATION (EM)-induced failure has become more and more serious due to the continuing increasing current densities in 5-nm technology and below. With the technology scaling to obtain high integrated density, reduction in interconnect cross section leads to large current densities. EM lifetime predicted by international roadmap for devices and systems (IRDSs) will be reduced by half as the technology node advances into a new generation [1]. As a result, an accurate and efficient EM assessment is critical for VLSI chip designs to meet the EM-induced reliability requirements.

Black equation was first proposed to predict EM-induced median-time-to-failure (MTF) [2]. However, the empirical equation can only work for one specific single interconnect since it is obtained by experimental data fitting. Then, Blech criterion is developed to check the immortality of one interconnect by comparing the product of current density and length with a specific threshold [3]. This method leads to conservative designs because it can not calculate transient hydrostatic stress and MTF. To predict EM-induced time-to-failure (TTF) accurately, several physics-based EM models and fast computational techniques have been proposed recently [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. Those EM models primarily employ the partial differential equation (PDE) which is called Korhonen's equations [16] to describe the transient hydrostatic stress in the general multisegment interconnects.

There are a number of traditional assessment techniques such as numerical methods and analytical methods which focus on solving the PDE efficiently and accurately [7], [11], [13], [14], [17], [18]. The mesh-based numerical methods, such as the finite element method [13] and finite difference method (FDM) [11], [17], require spatial and temporal discretization and cost lots of CPU time and memory. Therefore, analytical solutions have attracted much attention. Laplace transformation method [7], [19] and integral transform technique [20] can only handle simple interconnect structures, such as a single wire and a straight line multisegment interconnect. To mitigate the problem, a separation of variables (SOVs) method was proposed to perform an EM assessment of the general multisegment interconnect trees [14], [18]. However, this method suffers from the large time to compute eigenvalues.

Recently, deep learning-based approaches for solving PDEs [also called Scientific machine learning (ML)] have shown great promise in both accuracy and efficiency [21], [22]. Such ML-based methods provide new insight to perform EM assessments. However, as shown in recent works and our initial study, simply applying existing ML-based methods for

Manuscript received 20 July 2022; revised 3 January 2023 and 17 March 2023; accepted 17 April 2023. Date of publication 21 April 2023; date of current version 20 October 2023. This work was supported in part by NSF Grant under Grant OISE-1854276 and Grant CCF-2007135. This article was recommended by Associate Editor I. Vatajelu. *(Corresponding author: Sheldon X.-D. Tan.)*

The authors are with the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA 92521 USA (e-mail: stan@ece.ucr.edu).

Digital Object Identifier 10.1109/TCAD.2023.3269393

solving large engineering problems suffers convergence and low accuracy issues.

In this work, we propose an efficient physics-informed sparse regression (PISR) unsupervised learning framework with *linear time* complexity to solve Korhonen's equation for large-scale interconnects. Our key contributions are as follows.

1) We propose a novel PISR framework to solve the PDE describing EM-induced stress evolution dynamics in confined metal wires. The PISR method is a combination of linear regression and neural networks so that it can better model complicated dynamic systems while still maintaining regression efficiency. Unlike the physics-informed neural network (PINN), the PISR framework trains the trainable weights by using the Moore–Penrose generalized inverse algorithm, which is extremely faster than a widely used backpropagation algorithm.
2) The PISR method may sacrifice the accuracy to achieve good speedup over the PINN since the PISR method has fewer trainable weights and its structure is simpler. To mitigate this problem, we apply the domain decomposition method in the space. Specifically, for each wire segment, we define it as a subdomain, which is represented by one neural network. To reduce memory usage for each subdomain, we use different trainable weights but the same shared neural network to represent the subsolution.
3) We further apply sparse matrix techniques to store, multiply and invert matrices in the PISR method. So we call it sparse regression. The sparse matrix techniques accelerate the training speed of the PISR method and reduce its memory usage significantly. We show that the resulting PISR method has *linear* time complexity.
4) Last but not least, to further reduce the memory used for the training of large networks, we apply the time domain decomposition concept. Specifically, we divide the time period into many time intervals and apply the autoregressive model to compute the divided time intervals in sequence. In this way, we can solve each interval sequentially and separately, which can significantly reduce the memory usage of the PISR method due to the reduced time period in each training and thus be more scalable for large-scale interconnects.

Experimental results on different kinds of interconnect structures show that the proposed PISR method has the same accuracy level as the numerical methods. For a fair comparison to traditional methods, the computation time of the PISR method contains both training time and inference time. We show that the proposed PISR method indeed demonstrates true *linear* time complexity for general interconnect trees for the first time. Furthermore, PISR can achieve 8.9×, 20.6×, and 1284× speedup over the recently proposed semi-analytic method, called accelerated SOVs (ASOVs) method [14], FDM with model order reduction (FDM-MOR) [11], and FDM [23] for the interconnect with $N_T = 5000$, respectively. The speedup will increase further with the increasing number of segments. Furthermore, we show that PISR also achieves an 818× speedup in training over the plain PINN method which is based on the traditional backpropagation algorithm.

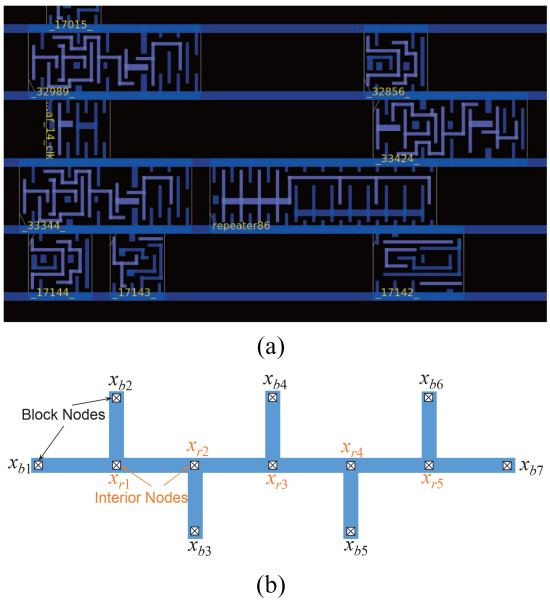


Fig. 1. (a) Metal-1 circuit layout from the "aes" design using nangate 45 nm in OpenROAD [24]. Several standard cell circuits are placed in the metal-1 power grid. As we can see, the power grid lines consist of several stubs, which can be seen as the interconnect tree. (b) General multisegment interconnect tree structure.

This article is organized as follows. Section II reviews the EM model and ML-based methods. Section III presents the novel PISR framework to solve Korhonen's equation describing hydrostatic stress evolution. Numerical results are presented in Section IV. Finally, Section V concludes this article.

II. Review of EM Model and ML-Based Methods

A. Physics-Based EM Modeling

EM is a physical phenomenon that the metal atoms move from one side to another side of the interconnects as conducting electrons driven by an applied electric field hit the atoms [3]. EM driving force creates tension at the cathode end and compression at the anode end of the wire. The lasting current increases the EM stress. When the stress reaches the critical level, the void is nucleated at the cathode end or the hillock is formed at the anode end of the line. As a result, the circuits are open or short at two ends, respectively, which are EM-induced reliability problems.

Korhonen's equation is well accepted to describe the hydrostatic stress evolution on the interconnect. Korhonen et al. [16] first proposed the PDEs model for a single wire. Fig. 1(a) shows a real chip power grid, which consists of many interconnect trees with stubs. Therefore, the general multisegment interconnect tree, as shown in Fig. 1(b), is an important interconnect structure in the power grid and needs to be studied. The other researchers further extend Korhonen's equation for the general multisegment interconnect tree which is continuously connected [7], [10], [18]. For a general multisegment interconnect tree with n nodes, including p interior junction nodes $x_r \in \{x_{r1}, x_{r2}, \ldots, x_{rp}\}$ and q block terminals $x_b \in \{x_{b1}, x_{b2}, \ldots, x_{bq}\}$, as shown in Fig. 1(b), the hydrostatic

stress distribution $\sigma(x,t)$ along the wire is described by the following Korhonen's equation [14], [18]:

$$\frac{\partial \sigma_{ij}(x,t)}{\partial t} = \frac{\partial}{\partial x}\left[\kappa_{ij}\left(\frac{\partial \sigma_{ij}(x,t)}{\partial x} + G_{ij}\right)\right], \ t > 0 \tag{1}$$

where ij denotes a branch connected to nodes i and j, n_r represents the unit inward normal direction of the interior junction node r on a branch ij, the value of which is $+1$ for the right direction and -1 for the left direction of the branch with an assumption of $x_i < x_j$, $G = -[(eZ^*\rho J)/\Omega]$ is the EM driving force, and $\kappa = [(D_a B\Omega)/k_B T]$ is the diffusivity of stress. $D_a = D_0 \exp(-E_a/k_B T)$, which is the effective atomic diffusion coefficient. D_0 is the preexponential factor, B is the effective bulk elasticity modulus, e is the electron charge, Z^* is the effective charge number, J is the current density, ρ is the resistivity of the interconnect, Ω is the atomic lattice volume, k_B is Boltzmann's constant, T is the absolute temperature, and E_a is the EM activation energy. The boundary conditions (BCs) block the atomic flux at block terminals x_b, which is expressed as

$$\text{BC}: \kappa_{bj}\left(\left.\frac{\partial \sigma_{bj}(x,t)}{\partial x}\right|_{x=x_b} + G_{bj}\right) = 0, t > 0. \tag{2}$$

The initial stress describes the thermal-induced residual stress along the interconnect at $t = 0$, which is given by

$$\text{IC}: \sigma_{ij}(x,0) = \sigma_{ij,T} \tag{3}$$

where σ_T is the initial thermal-induced residual stress. To connect each branch together, the stress and atomic flux should be continuous at interior nodes x_r. Therefore, the stresses at interior nodes x_r are the same

$$\begin{aligned}\text{SC}: \sigma_{rj_1}(x_r,t) &= \sigma_{rj_2}(x_r,t) \\ &= \cdots = \sigma_{rj_n}(x_r,t), t > 0\end{aligned} \tag{4}$$

where "SC" represents stress continuity conditions at the interior nodes x_r. n is the number of branches which is connected with node x_r. The atomic fluxes at the interior nodes x_r are conserved to zero

$$\text{AC}: \sum_{rj} \kappa_{rj}\left(\left.\frac{\partial \sigma_{rj}(x,t)}{\partial x}\right|_{x=x_r} + G_{rj}\right) \cdot n_r = 0, \ t > 0 \tag{5}$$

where "AC" represents atomic flux continuity conditions at the interior nodes x_r.

When the stress reaches a critical stress level σ_{crit}, the void is formed in the cathode node. The corresponding void BC [6], [25] for the cathode node is described as

$$\left.\frac{\partial \sigma_{ij}(x,t)}{\partial x}\right|_{x=x_l} \approx \frac{\sigma_{ij}(x_l,t)}{\delta} \tag{6}$$

where δ is the effective thickness of void boundary. In this article, δ is set to 1 nm. Based on the stress distribution on the wires, the void volume $V_v(t)$ is calculated by the atom conservation equation [6], [25]

$$V_v(t) = -\int_{\Omega_L} \frac{\sigma(t)}{B} dV + \int_{\Omega_L} \frac{\sigma_T}{B} dV \tag{7}$$

where Ω_L is the volume of all the connected interconnects. The resistance of the interconnect remains almost the same before the void volume reaches the critical volume. This is because the cross section of the via is not consumed by the void and the current can still flow through the copper [26].

When the void reaches critical volume V_{crit} or critical length L_{crit}, all current starts to flow through the high resistive barrier, which leads to a small resistance jump. The resistance change [5] can be estimated by

$$\Delta R(t) = \frac{V_v(t)}{WH}\left[\frac{\rho_{Ta}}{h_{Ta}(2H+W)} - \frac{\rho_{Cu}}{HW}\right] \tag{8}$$

where ρ_{Ta} and ρ_{Cu} are the resistivities of the barrier material (Ta/TaN) and copper, respectively, W and H are the width and the thickness of the interconnect, respectively, and h_{Ta} is the barrier layer thickness. As the increment of resistance is up to 10%, the interconnect is recognized as a failed wire since the IR drop fails to meet the requirement [6], [26].

When void's volume $V_v(t) = 0$, the EM failure process is called the nucleation phase. Once the void is formed, the EM failure process enters the post-voiding phase [6]. In the post-voiding phase, we can divide it into two subphases, which are the incubation phase and the growth phase [26]. In the incubation phase, the void starts to grow but the wire resistance does not change immediately. In the growth phase, the resistance starts to increase.

B. ML-Based Approaches for EM Analysis

Recently, several ML-based methods have been applied for EM analysis because of their fast prediction. These methods are divided into two strategies: 1) data-driven methods and 2) physics-informed methods. A data-driven method is a supervised learning method based on the dataset. Jin et al. [27] first employed a generative adversarial networks (GANs) method to predict EM stress by representing the interconnect structures with the image. It shows an order of magnitude speedup through comparison to the efficient analytic-based EM solver. However, this method can only predict the stress for a region with a fixed size because the output of the model is an image with a fixed size. Therefore, this model can not be used to predict stress for chips of the other size. The image is not an efficient way to represent the multisegment interconnects since the image has several blank regions. To efficiently describe the multisegment interconnect structure, a graph is leveraged to represent the node and edge of the interconnect trees. Therefore, a graph convolutional network (GCN)-based method is developed to estimate hydrostatic stress [28]. This ML-based model has a smaller model size and faster speed than the GAN-based method. However, those data-driven methods require a dataset with labeled data to train the model, which restricts their applications in real problems because labeled data generation is a big issue.

Physics-informed method, which is an unsupervised learning method [29], was proposed to tackle the problem of data generation. Based on automatic differentiation, a PINN [29] estimates the differential operators and adds the information of physics law, such as governing equations, BCs, and initial conditions, into the loss functions, which are used to

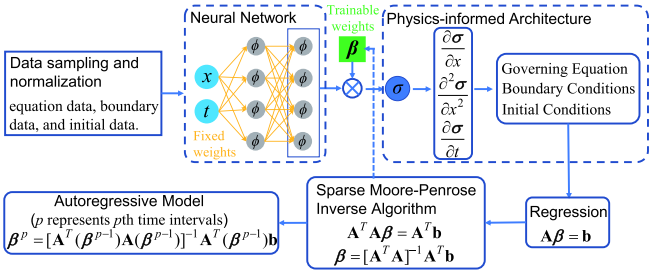


Fig. 2. PISR framework for EM analysis.

train the neural network by backpropagation without a dataset. Jin et al. [30] employed the PINN method to solve the electrostatic field with parameterization of voltage. Recently, Hou et al. have applied space-time PINN (STPINN) to compute EM stress evolution for the VLSI power grid [31]. They tried to add more neurons related to the predetermined collocation points. It can achieve better accuracy but takes a long time to train the model, which can be viewed as a tradeoff between accuracy and training time. Furthermore, STPINN only has been tested on multisegment interconnects with a small number of segments (<30) since it has convergence issues for large-scale interconnects. To mitigate the slow training speed of PINN, some researchers proposed to use shallow neural network instead of deep neural network, such as the physics-informed extreme learning method (PIELM) [32], local extreme learning machine (locELM) [33], Bernstein neural network (BeNN) [34], and physics-informed random project neural network (PIRPNN) [35]. However, these methods suffer from a large amount of memory usage, ill condition of the matrix, and $O(N^3)$ time complexity [36]. Therefore, applying unsupervised learning methods for EM assessment of large-scale interconnects remains a challenge.

III. PISR Method for EM Assessment

In this section, we propose a novel PISR framework to solve Korhonen's equation for general multisegment interconnect, as shown in Fig. 2. First, we sample enough points in equation domains, BCs and initial conditions. Then, we present a novel regression method based on PINN. Next, we use sparse matrix techniques to store, multiply, and solve the inverse of the sparse matrix so that the PISR algorithm is linear time complexity. Finally, we propose an autoregressive method to tackle the large time-scale problem and reduce memory significantly.

A. Data Sampling and Normalization

Like the PINN method, we need to sample enough data as training data to train the neural network. The sampling points, including initial data, boundary data and equation data, are generated by using normal sampling distribution. The PISR is a meshless method since the sampling points are not connected to form a mesh. Position x and time t from the sampling points are taken as inputs. Once we prepare the data, we have to normalize the training data, which is an important step in ML. In this article, we use standardization scaling, which is expressed as

$$\hat{X} = \frac{X - X_{\text{mean}}}{X_{\text{std}}} \tag{9}$$

where X_{mean} and X_{std} are mean values and standard deviations of the data X, respectively. X can be position x, time t, and stress σ.

With the scaling variables, Korhonen's equation (1)–(5) needs to be updated. Based on the derivative chain rule, we have

$$\frac{\partial \sigma}{\partial x} = \frac{\partial \sigma}{\partial \hat{\sigma}}\frac{\partial \hat{\sigma}}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial x} = \frac{\partial \hat{\sigma}}{\partial \hat{x}}\frac{\sigma_{\text{std}}}{x_{\text{std}}}, \quad \frac{\partial \sigma}{\partial t} = \frac{\partial \sigma}{\partial \hat{\sigma}}\frac{\partial \hat{\sigma}}{\partial \hat{t}}\frac{\partial \hat{t}}{\partial t} = \frac{\partial \hat{\sigma}}{\partial \hat{t}}\frac{\sigma_{\text{std}}}{t_{\text{std}}} \tag{10}$$

$$\frac{\partial^2 \sigma}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial \sigma}{\partial x}\right) = \frac{\partial}{\partial \hat{x}}\left(\frac{\partial \sigma}{\partial x}\right)\frac{\partial \hat{x}}{\partial x} = \frac{\partial^2 \hat{\sigma}}{\partial \hat{x}^2}\frac{\sigma_{\text{std}}}{x_{\text{std}}^2} \tag{11}$$

where $\hat{x}$, $\hat{t}$, and $\hat{\sigma}$ are the normalization of x, t, and σ, respectively, which make neural networks easy to be trained. x_{mean}, t_{mean}, and σ_{mean} are mean values of x, t, and σ at the sampling points, respectively. x_{std}, t_{std}, σ_{std} are standard deviations of x, t, and σ at the sampling points, respectively. Therefore, by substituting (10) and (11) into Korhonen's (1)–(5), we can obtain the normalized PDEs

$$\begin{aligned}
&\frac{\partial \hat{\sigma}_{ij}(\hat{x},\hat{t})}{\partial \hat{t}} - \frac{\kappa_{ij} t_{\text{std}}}{x_{\text{std}}^2}\frac{\partial \hat{\sigma}_{ij}^2(\hat{x},\hat{t})}{\partial \hat{x}^2} = 0\\
&\text{SC}: \hat{\sigma}_{rj_1}(\hat{x}_r,\hat{t}) - \hat{\sigma}_{rj_2}(\hat{x}_r,\hat{t}) = 0\\
&\text{AC}: \sum_{rj}\kappa_{rj} n_r \frac{\partial \hat{\sigma}_{rj}(\hat{x},\hat{t})}{\partial \hat{x}}\frac{\sigma_{\text{std}}}{x_{\text{std}}}\bigg|_{\hat{x}=\hat{x}_r} = -\sum_{rj}\kappa_{rj} n_r G_{rj}\\
&\text{BC}: \frac{\partial \hat{\sigma}_{bj}(\hat{x},\hat{t})}{\partial \hat{x}}\frac{\sigma_{\text{std}}}{x_{\text{std}}}\bigg|_{x=x_b} = -G_{bj}\\
&\text{IC}: \hat{\sigma}_{ij}(\hat{x},0)\sigma_{\text{std}} + \sigma_{\text{mean}} = \sigma_{ij,T}.
\end{aligned} \tag{12}$$

When the EM failure process enters the postvoiding phase, the void BC is normalized as

$$\frac{\partial \hat{\sigma}_{lj}(\hat{x},\hat{t})}{\partial \hat{x}}\frac{\sigma_{\text{std}}}{x_{\text{std}}}\bigg|_{x=x_l} \approx \frac{\hat{\sigma}_{lj}(x=x_l,\hat{t})\sigma_{\text{std}} + \sigma_{\text{mean}}}{\delta}. \tag{13}$$

After the normalization of data and PDEs, we can apply the regression method to solve this problem in the next section.

B. Regression Framework

In linear regression [37], the output is represented by a linear combination of inputs, which is expressed as

$$\hat{\sigma}(\hat{x},\hat{t}) = \begin{bmatrix} v_1 & v_2 \end{bmatrix}\begin{bmatrix} \hat{x} \\ \hat{t} \end{bmatrix} \tag{14}$$

where v_1 and v_2 are the weights to be determined by the linear regression. Although linear regression has a fast training speed, this method can not model nonlinear effects due to the linear function. To extend the linear regression for modeling more complex relationships, the neural network is added to the linear regression, which can be formulated by

$$\hat{\sigma}(\hat{x},\hat{t}) = \sum_{i=1}^{N_n} v_i \text{NN}_i(\hat{x},\hat{t}) = \boldsymbol{\alpha}^T\boldsymbol{\beta} \tag{15}$$

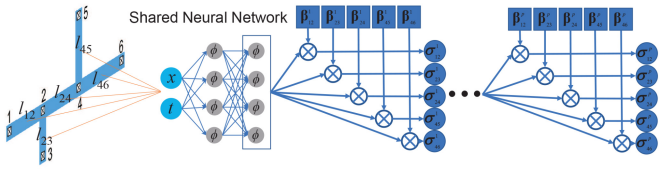


Fig. 3. Space-time domain decomposition method for regression framework. There are five segments and p time intervals in this example. Therefore, we use $5\times p$ trainable weights $\boldsymbol{\beta}_{ij}^{p}$ and one shared neural network $\mathrm{NN}(\hat{x},\hat{t})$ to represent the solutions.

where v_i is the coefficient, $\mathrm{NN}_i(\hat{x},\hat{t})$ is the ith outputs of neural network $\mathrm{NN}(\hat{x},\hat{t})$ with the inputs of normalized position $\hat{x}$ and time $\hat{t}$, and N_n is the number of outputs. The vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are written as

$$\begin{aligned}\boldsymbol{\alpha} &= \begin{bmatrix}\mathrm{NN}_1(\hat{x},\hat{t}) & \cdots & \mathrm{NN}_i(\hat{x},\hat{t}) & \cdots & \mathrm{NN}_{N_n}(\hat{x},\hat{t})\end{bmatrix}^T\\ \boldsymbol{\beta} &= \begin{bmatrix}v_1 & \cdots & v_i & \cdots & v_N\end{bmatrix}^T.\end{aligned} \tag{16}$$

$\mathrm{NN}(\hat{x},\hat{t})$ can be any kind of neural networks, such as fully connected networks [33], BeNN [34], Fourier neural network [38], radial basis function neural networks [35], and extreme learning machine (ELM) [32]. If $\mathrm{NN}(\hat{x},\hat{t})$ is a single layer, which is expressed as

$$\mathrm{NN}(\hat{x},\hat{t}) = \tanh\left(\mathbf{W}\begin{bmatrix}\hat{x}\\ \hat{t}\end{bmatrix}+\mathbf{b}\right) \tag{17}$$

expression (15) will become an ELM [39]. Note that the following derivation is not limited to ELM and can be applied for any kind of neural networks.

Then, we apply the linear combinations of neural network (15) to solve Korhonen's equations. By substituting (15) into (12), we have

$$\begin{aligned}&\lambda_g\left[\frac{\partial\boldsymbol{\alpha}_{ij}^T}{\partial\hat{t}}-\frac{\kappa_{ij}t_{\mathrm{std}}}{x_{\mathrm{std}}^2}\frac{\partial^2\boldsymbol{\alpha}_{ij}^T}{\partial\hat{x}^2}\right]\boldsymbol{\beta}_{ij}=0\\ &\lambda_s\begin{bmatrix}\boldsymbol{\alpha}_{rj_1}^T & -\boldsymbol{\alpha}_{rj_2}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\beta}_{rj_1}\\ \boldsymbol{\beta}_{rj_2}\end{bmatrix}=0\\ &\lambda_a\sum_{rj}\kappa_{rj}n_r\frac{\partial\boldsymbol{\alpha}_{rj}^T}{\partial\hat{x}}\frac{\sigma_{\mathrm{std}}}{x_{\mathrm{std}}}\bigg|_{\hat{x}=\hat{x}_r}\boldsymbol{\beta}_{rj}=-\lambda_a\sum_{rj}\kappa_{rj}n_rG_{rj}\\ &\lambda_b\frac{\partial\boldsymbol{\alpha}_{bj}^T}{\partial\hat{x}}\frac{\sigma_{\mathrm{std}}}{x_{\mathrm{std}}}\bigg|_{\hat{x}=\hat{x}_b}\boldsymbol{\beta}_{bj}=-\lambda_bG_{bj}\\ &\lambda_i\boldsymbol{\alpha}_{ij}^T\sigma_{\mathrm{std}}\boldsymbol{\beta}_{ij}=\lambda_i\left(\sigma_{ij,T}-\sigma_{\mathrm{mean}}\right).\end{aligned} \tag{18}$$

The derivative items $(\partial\boldsymbol{\alpha}_{ij}^T/\partial\hat{t})$, $(\partial^2\boldsymbol{\alpha}_{ij}^T/\partial\hat{x}^2)$, and $(\partial\boldsymbol{\alpha}_{ij}^T/\partial\hat{x})$ can be obtained easily by automatic differentiation of neural networks, which is the basic idea of PINN. Therefore, this regression method is based on the PINN method. The space domain decomposition method is employed to divide the whole interconnect tree into several single branches. In each branch ij, an individual trainable weight $\boldsymbol{\beta}_{ij}$ is used to represent stress $\hat{\sigma}_{ij}(\hat{x},\hat{t})$, as shown in Fig. 3. In the common interface, we enforce stress continuity and atom flux continuity. For the time domain, we also divide the whole time range into several small time intervals, as shown in Fig. 3. p represents the pth time interval. This idea is similar to the distributed PIELM (DPIELM) [32]. Note that $\mathrm{NN}(\hat{x},\hat{t})$ is a shared neural network, which can be employed for different branches. Therefore, we can just use one shared neural network and different trainable weights to represent the solution of each space and time subdomain. Our proposed idea of the shared neural network can reduce memory usage significantly. The vectors are represented by

$$\begin{aligned}\boldsymbol{\alpha}_{ij} &= \begin{bmatrix}\mathrm{NN}_1(\hat{x}_{ij},\hat{t}_{ij}) & \cdots & \mathrm{NN}_{N_n}(\hat{x}_{ij},\hat{t}_{ij})\end{bmatrix}^T\\ \boldsymbol{\beta}_{ij} &= \begin{bmatrix}v_{1,ij} & \cdots & v_{i,ij} & \cdots & v_{N,ij}\end{bmatrix}^T\end{aligned} \tag{19}$$

λ_g, λ_s, λ_a, λ_b, and λ_i are the weights to make values of all equations in the same order of magnitude, which can improve the condition number of the matrix. Similarly, to consider the postvoiding phase, we can obtain the void BC

$$\lambda_b\left[\frac{\partial\boldsymbol{\alpha}_{lj}^T}{\partial\hat{x}}\frac{\sigma_{\mathrm{std}}}{x_{\mathrm{std}}}\bigg|_{x=x_l}-\boldsymbol{\alpha}_{lj}^T\frac{\sigma_{\mathrm{std}}}{\delta}\bigg|_{x=x_l}\right]\boldsymbol{\beta}_{lj}\approx\lambda_b\frac{\sigma_{\mathrm{mean}}}{\delta}. \tag{20}$$

Once we get the loss function (18) and (20), we feed the sampling data, such as initial data, boundary data and equation data into the shared neural network. After that, the loss function can form a system of linear equations

$$\mathbf{A}\boldsymbol{\beta}=\mathbf{b} \tag{21}$$

where $\mathbf{A}$ is a $L\times(N_n\cdot N_{\mathrm{seg}})$ matrix, $\boldsymbol{\beta}$ is the coefficient to be determined, and $\mathbf{b}$ is an $L\times 1$ vector. L depends on the number of sampling points and branches. N_{seg} is the number of branches in the interconnect tree. Equation (21) is a linear regression problem, which can be solved by the linear least-squares methods.

In [33], they employed linear least square routine "lstsq" from the scipy.linalg package, which is a scientific library based on LAPACK. This routine computes the linear least squares problem (21) by using the singular value decomposition (SVD) of $\mathbf{A}$. The time complexity for computing SVD of an $m\times n$ matrix is $O(mn\min(n,m))$, which is very time-consuming and requires a large amount of memory. This routine "lstsq" can only be applied for the multisegment interconnect with less than 50 segments.

In [32], they use the Moore–Penrose generalized inverse of matrix $\mathbf{A}$ to solve the problem (21). $\mathbf{A}$ is not a square matrix. Therefore, both sides of (21) are multiplied by the transpose of $\mathbf{A}$, which is expressed as

$$\mathbf{A}^T\mathbf{A}\boldsymbol{\beta}=\mathbf{A}^T\mathbf{b}. \tag{22}$$

After that, matrix $\mathbf{A}^T\mathbf{A}$ is an $(N_n\cdot N_{\mathrm{seg}})\times(N_n\cdot N_{\mathrm{seg}})$ square matrix. They calculate the inverse of matrix $\mathbf{A}^T\mathbf{A}$ to obtain the final solution, which is expressed as

$$\boldsymbol{\beta}=\left[\mathbf{A}^T\mathbf{A}\right]^{-1}\mathbf{A}^T\mathbf{b}. \tag{23}$$

The total time of this method is determined by three parts. The first part is the time to generate the $\mathbf{A}$ matrix. The second part is the time of multiplication of two matrices $\mathbf{A}^T$ and $\mathbf{A}$. The third part is the time to compute the inverse of $\mathbf{A}^T\mathbf{A}$. The Moore–Penrose generalized inverse algorithm is faster than the linear least square routine "lstsq." However, in [32] and [36], they use the dense matrix to represent $\mathbf{A}$. Therefore, matrix $\mathbf{A}$ costs a large amount of memory due to lots of zero elements. Especially for DPIELM, the number of zero elements

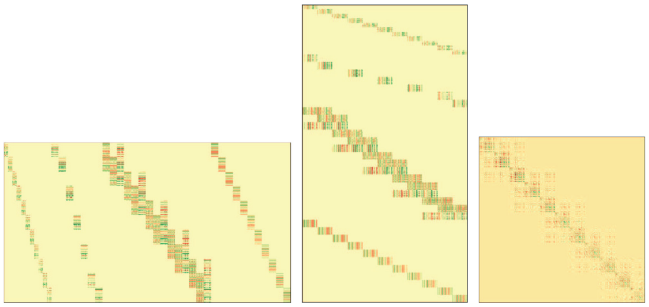


Fig. 4. (a) Transpose of matrix **A**. (b) Matrix **A**. (c) Matrix $\mathbf{A}^T\mathbf{A}$.

in matrix **A** increases dramatically with the increasing number of decomposed domains. What is worse, each domain is represented by one neural network. The total number of neural networks for the whole problem is very large so that this method consumes huge memory. Therefore, DPIELM can not be applied for large-scale interconnect trees of the chip due to the issue of large memory usage. It can be only applied for the multisegment interconnect with less than 150 segments. For the multiplication of $\mathbf{A}^T$ and **A**, there are thousands of operations to multiply zero elements, which are useless for the final results. The time complexity for the inverse of the dense matrix by using the Gauss–Jordan Elimination method is $O(N^3)$ [40]. Therefore, total computation time increases cubically as the number of segments becomes large.

C. Sparse Matrix Technique

To further improve the Moore–Penrose generalized inverse algorithm, we use the sparse matrix technique to represent matrix **A**. To demonstrate the time complexity of the proposed PISR algorithm, we use N_T T-junctions interconnect structure as an example, as shown in Fig. 1(b). We sample N_g equation data points, N_b boundary data points, N_{bi} continuity data points, and N_i initial data points as input data. The number of block terminals q is $N_T + 2$. The number of interior nodes p is N_T. The number of branches N_{seg} is $2N_T + 1$. The total number of sampling points is $N_{\text{points}} = (2N_g + 2N_i + N_b + 3N_{bi})N_T + N_g + N_i + 2N_b$. When we form matrix **A**, we use coordinate (COO) sparse format to store matrix **A**, as shown in Fig. 4(b). The number of nonzero elements in matrix **A** is estimated by

$$N_{\text{non-zero},\mathbf{A}} = \left[(2N_g + 2N_i + N_b + 7N_{bi})N_T + N_g + N_i + 2N_b\right]N_n. \tag{24}$$

Then, we use the sparse multiplication routine "mm" from torch.sparse package to compute $\mathbf{A}^T\mathbf{A}$ to avoid multiplying zero elements. The number of multiplication operations of nonzero elements is calculated by

$$N_{\text{mul}} = \left[(2N_g + N_b + 2N_i + 17N_{bi})N_T + N_g + 2N_b + N_i\right]N_n^2. \tag{25}$$

The matrix $\mathbf{A}^T\mathbf{A}$ is also a sparse matrix, which can be solved by the sparse solver routine "spsolve" from scipy.sparse.linalg package. The number of nonzero elements in $\mathbf{A}^T\mathbf{A}$ is given by

$$N_{\text{non-zero},\mathbf{A}^T\mathbf{A}} = (8N_T + 1)N_n^2. \tag{26}$$

Therefore, based on (24)–(26), the total time is proportional to the number of T-junctions N_T, which means that the time complexity of the proposed method is $O(N)$. The linear time complexity can be demonstrated in Section IV-D. Based on the sparse matrix technique, we can speed up the solving process significantly and reduce memory tremendously. In this method, we only need to calculate the inverse of $\mathbf{A}^T\mathbf{A}$, which is an $(N_n \cdot (2N_T + 1)) \times (N_n \cdot (2N_T + 1))$ matrix. Therefore, this method does not depend on the number of nodes (N_{points}), but depends on the number of the coefficient $(N_n \cdot (2N_T + 1))$. The finite element method must require mesh and depend on (N_{points}), which is larger than $(N_n \cdot (2N_T + 1))$. As a result, the proposed PISR method can reduce the dimensionality of the data from (N_{points}) to $(N_n \cdot (2N_T + 1))$ and is more efficient than the finite element method.

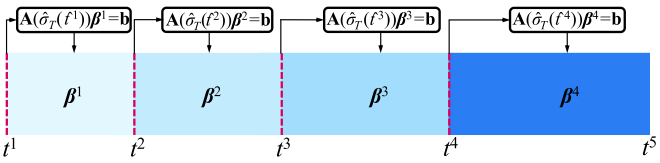


Fig. 5. Autoregressive model, which computes the solution $\boldsymbol{\beta}^p$ at the pth time interval based on the previous solution $\boldsymbol{\beta}^{p-1}$ at the $(p-1)$th time interval.

D. Autoregressive Model

In the time domain, we can divide the time into several small time intervals, as shown in Fig. 5. This adaptive time decomposition method can improve the accuracy of PISR with fewer neurons N_n. The time interval is smaller as the rate of change of the stress is larger. In general, we can divide the time range into three time intervals to balance the accuracy and efficiency of PISR. Then, we have different coefficients for each time interval, which are expressed as

$$\boldsymbol{\beta}(t) = \begin{cases} \boldsymbol{\beta}^1, & t^1 < t < t^2 \\ \boldsymbol{\beta}^2, & t^2 < t < t^3 \\ \cdots & \\ \boldsymbol{\beta}^p, & t^p < t < t^{p+1} \\ \cdots & \end{cases} \tag{27}$$

where p represents the pth time interval. t^p is the initial time for the pth time intervals. The initial conditions for pth time interval $(p > 1)$ can be updated by results from the $(p-1)$th time interval, which is given by

$$\hat{\sigma}_T(\hat{x}, \hat{t}^p) = \sum_{i=1}^{N} v_i^{p-1}\text{NN}_i(\hat{x}, \hat{t}^p) = \boldsymbol{\alpha}^T\boldsymbol{\beta}^{p-1}. \tag{28}$$

Different from DPIELM [32], this autoregressive model predicts the stress in a time of $t^p < t < t^{p+1}$ based on its previous values in a time of $t^{p-1} < t < t^p$, which can be represented by

$$\boldsymbol{\beta}^p = \left[\mathbf{A}^T\left(\boldsymbol{\beta}^{p-1}\right)\mathbf{A}\left(\boldsymbol{\beta}^{p-1}\right)\right]^{-1}\mathbf{A}^T\left(\boldsymbol{\beta}^{p-1}\right)\mathbf{b} \tag{29}$$

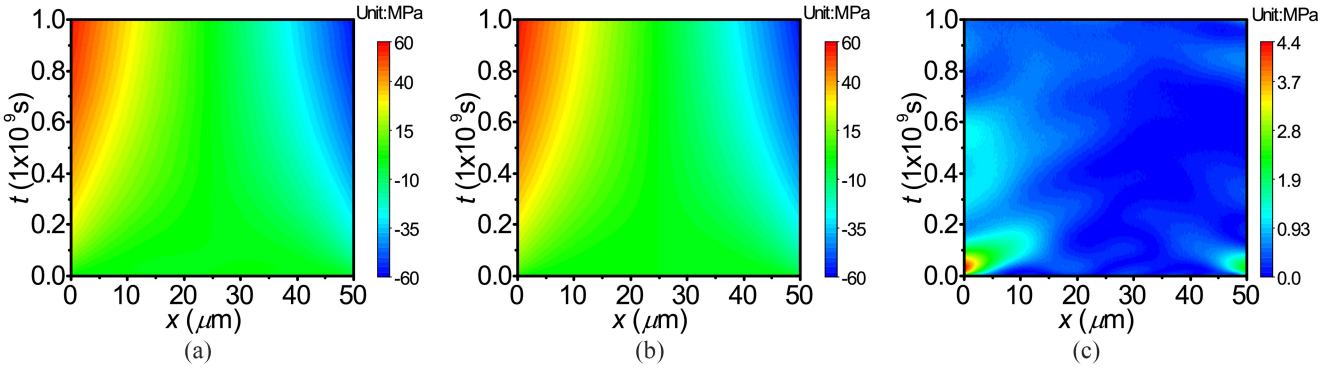


Fig. 6. Comparison of the proposed PISR results with the solution obtained by using COMSOL for a single wire. (a) PISR solution $\sigma_{\text{PISR}}(x, t)$. (b) COMSOL solution $\sigma_{\text{COMSOL}}(x, t)$. (c) Absolute error $|\sigma_{\text{PISR}} - \sigma_{\text{COMSOL}}|$. The neural network is a single hidden layer with $N_n = 90$ neurons, and the learnable weights are randomly generated in the range of [–1, 1]. λ_g, λ_b, and λ_i are set to 10^2, 10^{-11}, and 3.4×10^{-6}, respectively. σ_{std} and σ_{mean} are set to 10^9 and 0, respectively.

where $\mathbf{A}(\boldsymbol{\beta}^{p-1})$ means $\mathbf{A}$ matrix is updated by the initial condition $\hat{\sigma}_T(\hat{t}^p)$ or $\boldsymbol{\beta}^{p-1}$, as shown in Fig. 5. Therefore, this method can solve each time interval step by step and costs less memory than DPIELM since DPIELM solves several time intervals simultaneously. Based on the autoregressive model, the PISR method can further reduce memory usage. Finally, the proposed PISR method can be applied for the multisegment interconnect with more than $N_T = 5000$, which is demonstrated in Section IV-D.

IV. Numerical Results and Discussion

We demonstrate the accuracy and scalability of our proposed PISR method on several experimental results. The commercial software COMSOL is first used to validate the accuracy of our proposed PISR method for two kinds of interconnects: 1) straight line multisegment interconnects and 2) general multisegment interconnect tree. Next, we randomly generate benchmarks with N_T T-junctions interconnect tree to illustrate the $O(N)$ time complexity of the proposed PISR method. For a fair comparison to traditional methods, such as FDM [23], FDM with model order reduction (FDM-MOR) [11], and an analytical method which is called ASOV method [14], the computation time of the PISR method contains both training and inference time.

The PISR method is implemented in PyTorch platform. A scientific library, called scipy.sparse package, is employed to solve the inverse of the sparse matrix. The PISR method is trained and tested on a Linux server with 2 Xeon E5-2699v4 2.2-GHz processors and 315-GB RAM. The parameters [19], [41] used in EM model are: $Z^* = 1$, $e = 1.609 \times 10^{-19}$ C, $E_a = 0.83e$ V, $\rho_{\text{Cu}} = 1.95 \times 10^{-8}$ $\Omega\cdot$ m, $B = 28$ GPa, $\Omega = 1.182 \times 10^{-29}$ m^3, $T = 373$ K, and $\sigma_{\text{crit}} = 41$ MPa.

A. Single Wire

We use a single wire to illustrate the whole flow of the PISR method. We first prepare the training data, as shown in Fig. 7, which consists of equation data, boundary data and initial data. The numbers of sampling points in interior domain, BCs and initial conditions are $N_g = 300$, $N_b = 120$, and $N_i = 60$, respectively. The normalized parameters x_{std}, x_{mean}, t_{std}, and

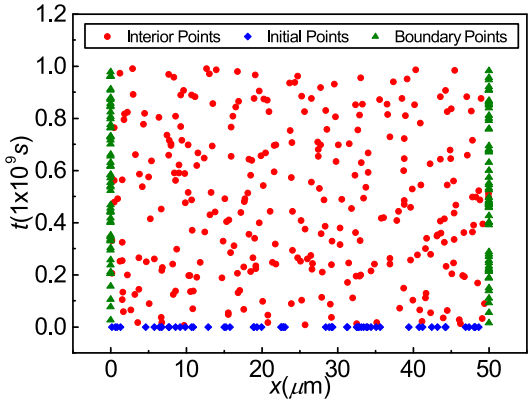


Fig. 7. Data sampling in interior domain, BCs and initial conditions. The numbers of sampling points in interior domain, BCs, and initial conditions are $N_g = 300$, $N_b = 120$, and $N_i = 60$, respectively.

t_{mean} are obtained from the sampling data. σ_{std} and σ_{mean} are set to 10^9 and 0, respectively, because stress σ is unknown and needs to be determined. Next, training data is taken as inputs and the least-squares problem is formed by using the normalized Korhonen equation. Finally, the Moore–Penrose inverse algorithm is employed to train this network and obtain the weights $\boldsymbol{\beta}$. For this simple wire, we do not need domain and time decomposition.

Fig. 6 shows the stress maps (0–10^9 s) which are estimated by the proposed PISR method and COMSOL. The current density is 1×10^{10} A/m^2. The length of the single wire is 50 μm. As we can see, the stress map from the PISR has a good agreement with that of COMSOL. Based on Fig. 6(c), the maximum relative error max $|\sigma_{\text{PISR}} - \sigma_{\text{COMSOL}}|/|\sigma_{\max} - \sigma_{\min}|$ is 3.9%, which occurs in early time. This is because stress changes quickly so that the single layer cannot fit well. We pay more attention on stress in late time. The relative error for late time is below 0.8%.

B. Straight Line Multisegment Interconnect Example

The second example is a straight line multisegment interconnect with ten segments, as shown in Fig. 9, which is the key component of power grid benchmarks [19]. The

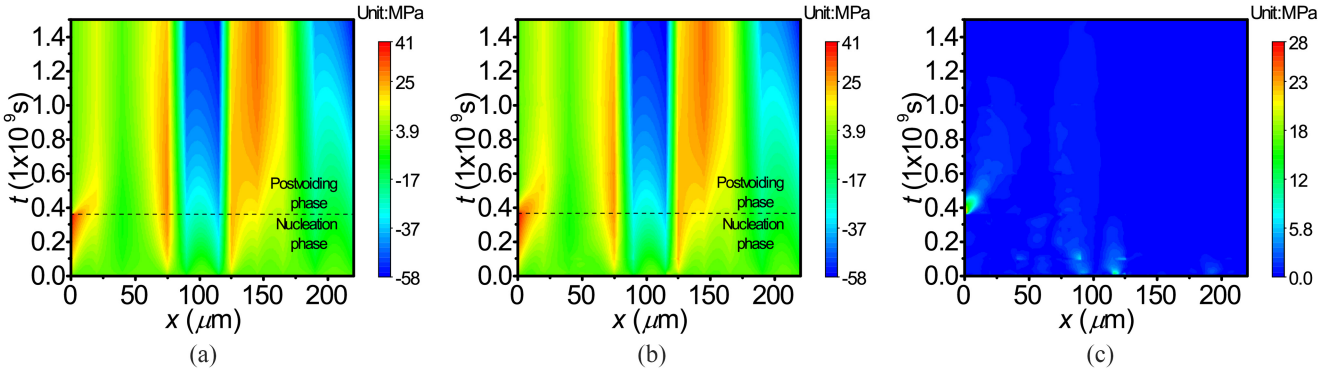


Fig. 8. Comparison of the proposed PISR results with the solution obtained by using COMSOL for the straight line multisegment interconnect with ten segments. (a) PISR solution $\sigma_{\text{PISR}}(x, t)$. (b) COMSOL solution $\sigma_{\text{COMSOL}}(x, t)$. (c) Absolute error $|\sigma_{\text{PISR}} - \sigma_{\text{COMSOL}}|$. The numbers of sampling points in interior domain, continuity conditions, BCs, and initial conditions for each subdomain are $N_g = 3 \times 10^6 \cdot L$, $N_{bi} = 30$, $N_b = 30$, and $N_i = 30$, respectively. L is the length of one segment. The neural network is a single hidden layer with $N_n = 90$ neurons, and the learnable weights are randomly generated in the range of [−1, 1]. λ_g, λ_s, λ_a, λ_b, and λ_i are set to 10^2, 3.4×10^{-6}, 0.9×10^{-11}, 0.9×10^{-11}, and 3.4×10^{-5}, respectively. σ_{std} and σ_{mean} are set to 10^9 and 0, respectively.

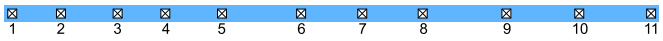


Fig. 9. Straight line multisegment interconnect with ten segments.

TABLE I
PARAMETERS FOR THE STRAIGHT LINE MULTISEGMENT INTERCONNECT WITH TEN SEGMENTS

Brch#	J (A/m^2)	$L(\mu$m)	Brch#	J (A/m^2)	$L(\mu$m)
$l_{1,2}$	-1×10^{10}	20	$l_{6,7}$	2×10^{10}	10
$l_{2,3}$	-1.5×10^{10}	20	$l_{7,8}$	-0.5×10^{10}	20
$l_{3,4}$	-1×10^{10}	35	$l_{8,9}$	-1×10^{10}	20
$l_{4,5}$	-3×10^{10}	15	$l_{9,10}$	-1.5×10^{10}	25
$l_{5,6}$	-1×10^{10}	25	$l_{10,11}$	-0.5×10^{10}	30

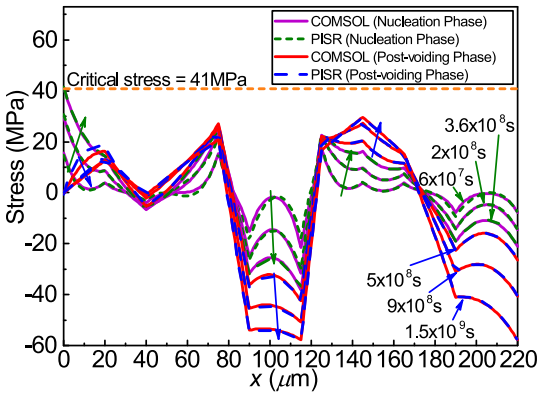


Fig. 10. Stress distribution on the ten-segment interconnect with different time points for nucleation and postvoiding phases.

current densities and lengths of ten segments are described in Table I. For this structure, both nucleation and postvoiding phases are simulated by using the proposed PISR method. Due to the limited representation capability of few trainable weights of PISR method, we leverage the domain decomposition method for space and time to improve the accuracy of the PISR method. Each segment ij is represented by the trainable weights $\boldsymbol{\beta}_{ij}$. But all segments share one neural network. The whole time range ($0 \sim 1.5 \times 10^9$ s) is divided into six time intervals, such as [0, 10^7 s], [10^7 s, 10^8 s], [10^8 s, 3.6×10^8 s], [3.6×10^8 s, 6×10^8 s], [6×10^8 s, 1×10^9 s], and [1×10^9 s, 1.5×10^9 s]. By applying the autoregressive model for time decomposition, we can reduce memory usage by 5/6 = 83% compared with DPIELM.

The maximum stress on the multisegment interconnect reaches the critical stress (41 MPa) at $t = 3.6 \times 10^8$ s, as shown in Figs. 8 and 10. Fig. 8 shows the stress maps ($0 \sim 1.5 \times 10^9$ s) of ten-segment interconnect with nucleation and postvoiding phases obtained from the proposed PISR method and COMSOL. As we can see, the proposed PISR method shows a perfect match with COMSOL for both two phases. To further illustrate the accuracy, we plot the stress distribution on the ten segments with different time points in Fig. 10. It can be seen that the PISR method matches COMSOL perfectly.

To demonstrate the efficiency of the Moore–Penrose generalized inverse algorithm, we also apply a plain PINN for EM nucleation phase analysis of the ten segments by using a 7-layer multilayer perceptron. Adam optimizer was employed to train the model and its learning rate was set to 10^{-4}. The training time (2.2 s) of the PISR achieves 818× speedup over that (1800 s) of the plain PINN. What is more, the relative error of PISR method for 1.5×10^9 s is 1.1%, which is much better than the accuracy of the plain PINN (10.7%). Note that we did not compare our method with [31], as this method can be essentially viewed as a plain PINN method with some tradeoffs between training accuracy and training time due to more complicated neuron representations.

Based on the transient stress obtained by the PISR model, we calculate the void length by (7), as shown in Fig. 11. The maximum stress increases in the nucleation phase and the void length is zero. When the maximum stress reaches the critical stress (41 MPa), the void is formed in 11.4 years ($t = 3.6 \times 10^8$ s) and the stress at the void boundary is released to zero. We can use the bisection algorithm to find the time to reach critical stress. After that, the void starts to grow in the postvoiding phase. However, the resistance does not change until the void length exceeds the

TABLE II
COMPARISON BETWEEN OUR PROPOSED PISR METHOD AND COMSOL ON OPENROAD BENCHMARKS WITH TWO PROCESS TECHNOLOGIES

Tech	Design	#Lines	#Segments	Steady-state (lines)		EM-void	PISR	COMSOL	Speedup
				Immortal	Mortal	(t=20 yrs)	(seconds)	(seconds)	
nangate 45 nm	aes	402	132276	199	203	18	5118	20424	4×
	gcd	34	968	27	7	0	17.4	35.3	2×
sky-hd 130 nm	aes	574	159615	553	21	0	114	220	1.9×
	gcd	91	4042	91	0	0	0	0	0
	ibex	984	469152	939	45	0	1236	8352	6.8×

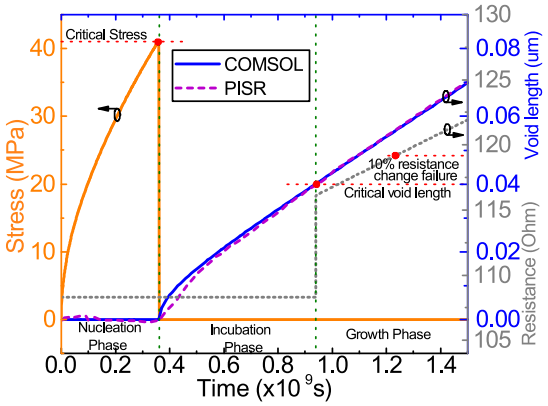


Fig. 11. Evolution of stress, void length, and resistance for both nucleation phase and postvoiding phase. The width, thickness, and length of interconnect are 40 nm, 90 nm, and 20 μm, respectively. The electrical conductivity of barrier material (Ta/TaN) is 1.76×10^{-7} Ω·m. The barrier layer thickness is 4 nm.

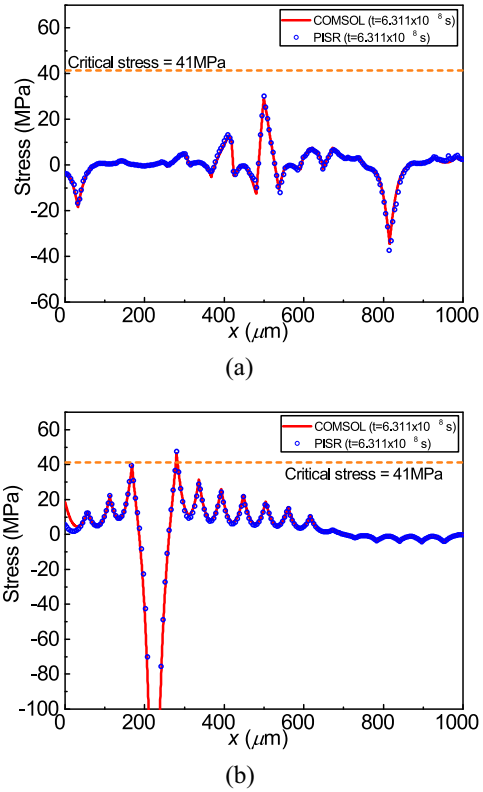
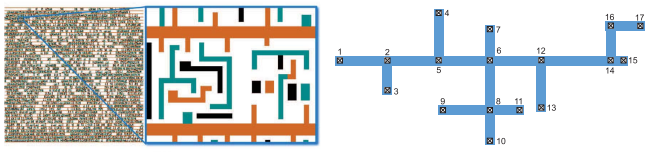


Fig. 12. Stress distributions on the straight line multisegment interconnect with (a) 371 segments and (b) 19 segments at $t = 20$ years. The two interconnects of length 1000 μm are obtained from the aes benchmark with nangate 45-nm technology in OpenROAD. The 371-segment and 19-segment lines are located in the metal-1 layer and metal-7 layer, respectively. The numbers of sampling points in interior domain, continuity conditions, BCs and initial conditions for each subdomain are $N_g = 3 \times 10^6 \cdot L$, $N_{bi} = 30$, $N_b = 30$, and $N_i = 30$, respectively. L is the length of one segment. The neural network is a single hidden layer with $N_n = 30$ neurons, the learnable weights are randomly generated in the range of [–1, 1]. λ_g, λ_s, λ_a, λ_b, and λ_i are set to 10^2, 5×10^{-6}, 0.9×10^{-11}, 0.8×10^{-11}, and 3.4×10^{-5}, respectively. σ_{std} and σ_{mean} are set to 10^9 and 0, respectively.

critical length 0.04 μm (the diameter of the via). Therefore, we can divide the postvoiding phase into the incubation phase and the growth phase. Based on (8), the resistance of the wire starts to change in 29.8 years ($t = 9.4 \times 10^8$ s). The wire fails in 39 years ($t = 1.23 \times 10^9$ s) since the resistance of the wire increases by 10% [6], [26]. The shape of resistance over time is similar to the experimental results in [42], which demonstrates that the physics-based EM model is reasonable. As we can see, the void length obtained by our proposed PISR method agrees well with the results from COMSOL.

To show the application of PISR on real power grid benchmarks, we use the OpenROAD tool [24] to automatically generate aes, gcd, and ibex circuits with nangate 45-nm and sky 130-nm technology since they are the test cases in OpenROAD. The currents for the power grids are computed by PDNSim [43]. The power grid consists of many straight line multisegment interconnect, as shown in Fig. 9. First, we use the fast immortality check algorithm [44] to check the immortal and mortal wires. Second, we use our proposed PISR method to perform transient analysis for the mortal wires to check the void at 20 years. If the stress exceeds the critical stress at 20 years, then these multisegment wires are marked as EM-void ($t = 20$ years). The results on OpenROAD benchmarks are illustrated in Table II. As we can see, the number of interconnects with 130-nm technology for EM-void ($t = 20$ years) is zero. This is reasonable because interconnects with 130-nm technology have a smaller current and larger width/thickness compared with 45 nm. The number of gcd circuit interconnects with 45-nm technology for EM-void ($t = 20$ years) is also zero since gcd is a small circuit with a short interconnect. The shorter length leads to better EM reliability. To compare the efficiency between PISR and COMSOL, we only compute the runtime of transient analysis. PISR can achieve a large speedup (6.8×) over COMSOL for large cases because PISR is a linear time algorithm which shows great advantage in large problem. We did not compare PISR with the state-of-the-art SOVs method on these power grid benchmarks because the SOV method is really fast for straight line multisegment interconnect. However, SOV is slower than PISR in the cases which have tree structures and a large number of segments, which will be demonstrated in Section IV-D. Fig. 12



(a) (b)

Fig. 13. (a) Real chip layout with standard cell synthesis and placement from [45]. The VDD and VSS lines have short stubs, which provide the power for the standard cells. The stubs cannot be ignored because the distance between two adjacent stubs is as long as the length of stub. (b) General multisegment interconnect tree structure which considers the short stubs.

TABLE III
PARAMETERS FOR THE GENERAL MULTISEGMENT INTERCONNECT TREE

Brch#	J (A/m^2)	$L(\mu$m)	Brch#	J (A/m^2)	$L(\mu$m)
$l_{1,2}$	-3×10^{10}	30	$l_{10,8}$	2×10^{10}	20
$l_{3,2}$	-0.5×10^{10}	20	$l_{8,11}$	1×10^{10}	20
$l_{2,5}$	-1×10^{10}	30	$l_{6,12}$	1×10^{10}	30
$l_{5,4}$	2×10^{10}	30	$l_{13,12}$	-0.5×10^{10}	30
$l_{5,6}$	-1×10^{10}	30	$l_{12,14}$	-1.5×10^{10}	40
$l_{6,7}$	1.5×10^{10}	20	$l_{14,15}$	-1×10^{10}	10
$l_{8,6}$	1×10^{10}	30	$l_{14,16}$	-0.5×10^{10}	20
$l_{9,8}$	3×10^{10}	30	$l_{16,17}$	-1×10^{10}	20

shows the stress distributions on the straight line multisegment interconnect with Fig. 12(a) 371 segments and Fig. 12(b) 19 segments at $t = 20$ years, which are obtained from the aes benchmark with nangate 45-nm technology in OpenROAD. The 371-segment and 19 segments with the same length 1000 μm are located in the metal-1 layer and metal-7 layer, respectively. As we can see, the transient analysis shows that The 371-segment line does not form a void at 20 years, but 19-segment line has a void at 20 years. The results obtained from PISR and COMSOL are matched well.

C. General Multisegment Interconnect Tree Example

Fig. 13(a) shows a real chip layout with standard cell synthesis and placement, which comes from [45]. As we can see, the VDD and VSS lines have short stubs, which provide the power for the standard cell. Note that we can not ignore short stubs since the distance between two adjacent stubs is as long as the length of the stubs. If we consider the stubs, the number of segments increases dramatically, which brings a challenge for EM analysis. We apply the proposed PISR method to perform EM analysis for a general multisegment interconnect tree with several stubs, as shown in Fig. 13(b). The current density and length for each branch are shown in Table III. Then, the domain decomposition method for time and space is also employed to improve the accuracy of the PISR method. Each branch ij is represented by the trainable weights $\boldsymbol{\beta}_{ij}$. The time range ($0 \sim 2\times10^9$ s) is divided into three time intervals, which are [0, 10^8 s], [10^8 s, 10^9 s], and [10^9 s, 2×10^9 s].

Fig. 14 shows the transient stress distribution of the general multisegment interconnect tree at different time points. As we can see, the stress obtained by the proposed PISR method coincides exactly with the results from COMSOL. The relative error of the PISR method for 2×10^9 s is 0.34%, which is the same accuracy level as the numerical methods.

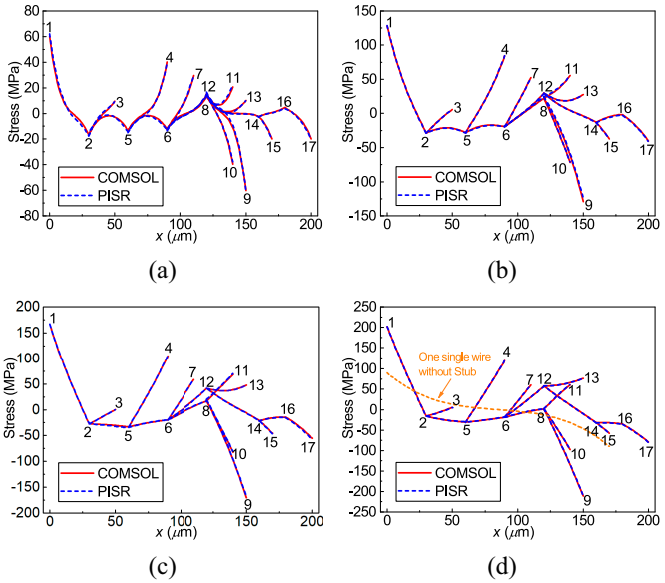


Fig. 14. Transient stress distribution of a multisegment interconnect tree at (a) $t = 1\times10^8$ s, (b) $t = 5\times10^8$ s, (c) $t = 1\times10^9$ s, and (d) $t = 2\times10^9$ s. The numbers of sampling points in interior domain, continuity conditions, BCs, and initial conditions for each subdomain are $N_g = 3\times10^6\cdot L$, $N_{bi} = 30$, $N_b = 30$, and $N_i = 30$, respectively. L is the length of one segment. The neural network is a single hidden layer with $N_n = 30$ neurons, and the learnable weights are randomly generated in the range of [–1, 1]. λ_g, λ_s, λ_a, λ_b, and λ_i are set to 10^2, 2×10^{-6}, 1×10^{-11}, 1×10^{-11}, and 3.4×10^{-6}, respectively. σ_{std} and σ_{mean} are set to 10^9 and 0, respectively.

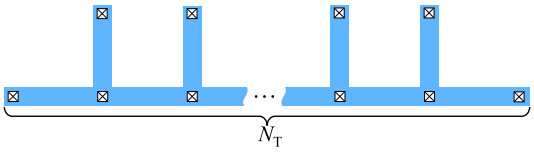


Fig. 15. Interconnect tree benchmark with N_T T-junctions.

To study the impact of stubs, we also calculate the stress on the VDD or VSS line without stubs, as shown in Fig. 14(d) (orange dotted–dashed line). This is a single wire with a length 170 μm and a current density 1×10^{10} A/m^2. It can be observed that the maximum stress without stubs is smaller than that of simulation with stubs. Therefore, to model EM accurately, we need to consider the stubs. In power grid benchmarks of OpenROAD, power grids only consist of the straight line multisegment interconnect, as shown in Fig. 9. Such power grids do not consider specific standard cell circuits and ignore the stubs. Therefore, the multisegment interconnect tree in Fig. 13(b) is viewed as a single wire, which is one element of a straight line multisegment interconnect. By considering the stubs, we calculate the hydrostatic stress for a general multisegment interconnect tree with 16 segments. The number of segments with stubs is 16× larger than that of a single wire without stubs.

D. Comparison With Existing Numerical Methods

In this section, we compare the proposed PISR method against some of the published EM numerical and semi-analytic methods over a number of interconnect trees ranging from $N_T = 100$ to $N_T = 5000$, as shown in Fig. 15. We compare

TABLE IV
COMPARISON BETWEEN OUR PROPOSED PISR METHOD AND THE OTHER TRADITIONAL METHODS

N_T	PISR	FDM [23]	FDM-MOR [11]	ASOV [14]	PISR Speedup w.r.t.		
	t_{PISR} (s)	t_{FDM} (s)	t_{MOR} (s)	t_{ASOV} (s)	FDM	FDM-MOR	ASOV
100	23.2	37.4	0.406	0.33	1.6×	0.017×	0.014×
200	37.8	205	2.60	1.18	5.4×	0.069×	0.031×
400	75.1	1519	20.2	4.73	20.2×	0.27×	0.063×
500	99.3	2376	34.1	7.97	23.9×	0.34×	0.08×
750	172	7609	108	22.4	44.2×	0.63×	0.13×
1000	197	1.53E4	253	51.2	77.7×	1.28×	0.26×
1250	241	2.88E4	460	142	119.6×	1.91×	0.59×
1500	291	5.21E4	791	314	179×	2.71×	1.08×
1750	350	8.05E4	1240	576	230×	3.54×	1.65×
2000	417	1.11E5	1815	862	266×	4.35×	2.07×
3000	616	3.64E5	6023	2764	591×	9.78×	4.49×
4000	1027	8.61E5	1.42E4	6295	838×	13.8×	6.13×
5000	1347	1.73E6	2.77E4	12062	1284×	20.6×	8.95×

our PISR against the recently proposed *FDM* method [23], the FDM with model order reduction method, *FDM-MOR* method [11], and the recently proposed semi-analytic EM analysis method, *ASOV* [14] as shown in Table IV. Note that we run ASOV code on a Linux server with two Xeon E5-2699v4 2.2-GHz processors and 315-GB RAM, which is the same as the platform where PISR is tested. We also show the speedup over the three methods in columns from 5 to 7 in Table IV.

For our *PISR*, the time reported in the table also consists of a few components: The training time t_L for the proposed PISR method consists of three parts as mentioned in Section III-B. They are the time t_A to generate the $\mathbf{A}$ matrix, the time t_M to multiply $\mathbf{A}^T$ and $\mathbf{A}$, and the time t_S to compute the inverse of $\mathbf{A}^T\mathbf{A}$. We have the relationship $t_L = t_A + t_M + t_S$. The inference time t_I is to predict the testing set with uniform sampling points. The total time t_T of the proposed PISR method is the summation of t_L and t_I. Fig. 16(a) shows the several time components for different computing parts of the proposed PISR method. The majority of computational time is t_A, which can be improved in the future. As we can see, all time components increase linearly with the increasing number of segments, which demonstrates that the proposed PISR method has $O(N)$ time complexity as mentioned in Section III-C.

To further compare with other published methods intuitively, we present time data with the log-log plot, as shown in Fig. 16(b). The time for the proposed PISR method consists of training time t_L and inference time t_I. We use the linear line to fit the data. The slope of the line represents the empirical runtime trend, which is computed by $(\log(T_1) - \log(T_2))/(\log(N_1) - \log(N_2))$ where T_1 and T_2 denote the time, and N_1 and N_2 represent the number of segments.

The empirical runtime trend of the proposed PISR method is $O(N)$, which is the same as the time complexity analysis in Section III-C. However, the empirical runtime trend for the other traditional methods is about $O(N^{2.7})$, as shown in Fig. 16(b). For the interconnect with $N_T = 5000$, the proposed PISR method achieves **8.9**×, **20.6**×, and **1284**× speedup over ASOV, FDM-MOR, and FDM, respectively. This is a clear trend that as the number of segments increases, the speedup will also increase. It indicates the truly scalable of the proposed method as it has a linear time complexity.

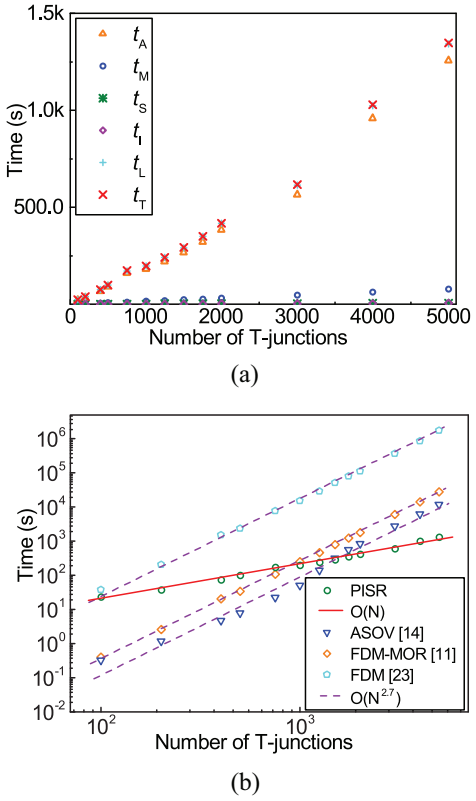


Fig. 16. (a) Time components of different parts for the proposed PISR method. (b) Comparison of the proposed PISR method, FDM, FDM-MOR, and ASOV. The length for each branch is 30 μm. The numbers of sampling points in interior domain, continuity conditions, BCs and initial conditions for each subdomain are $N_g = 3 \times 10^6 \cdot L$, $N_{bi} = 30$, $N_b = 30$, and $N_i = 30$, respectively. L is the length of one segment. The time range ($0 \sim 2 \times 10^9$ s) is divided into three time intervals, which are [0, 10^8 s], [10^8 s, 10^9 s], and [10^9 s, 2×10^9 s]. The neural network is a single hidden layer with $N_n = 30$ neurons, and the learnable weights are randomly generated in the range of [–1, 1]. λ_g, λ_s, λ_a, λ_b, and λ_i are set to 10^2, 2×10^{-6}, 1×10^{-11}, 1×10^{-11}, and 3.4×10^{-6}, respectively. σ_{std} and σ_{mean} are set to 10^9 and 0, respectively.

V. CONCLUSION

In this article, we proposed a novel PISR framework to solve Korhonen's equations for general multisegment wires. The PISR method trains the trainable weights through the Moore–Penrose generalized inverse algorithm, which is extremely faster than the backpropagation algorithm. To

improve the accuracy of PISR for complex multisegment interconnects, we employed domain decomposition schemes in both space and time. Our proposed idea of the shared neural network can reduce memory cost significantly. Furthermore, we propose to use sparse matrix techniques to accelerate the training speed of the PISR method and reduce its memory usage significantly. Finally, an autoregressive model is employed to simulate the divided time intervals in sequence to further reduce memory cost. The numerical results on different kinds of interconnect structures show that the proposed PISR method has the same accuracy level as the numerical methods. The numerical results on T-junctions interconnect trees show that the proposed PISR method indeed demonstrates linear time complexity for general interconnect trees for the first time. Furthermore, PISR can deliver 8.9×, 20.6×, and 1284× speedups over the recently proposed semi-analytic method (ASOV), FDM accelerated with model order reduction (FDM-MOR), and FDM for the interconnect with $N_T = 5000$, respectively. Furthermore, we show that PISR also achieves an 818× speedup in training speed over the plain PINN method based on the traditional backpropagation algorithm.

REFERENCES

[1] "IEEE international roadmap for devices and systems (IRDS)." 2021. [Online]. Available: https://irds.ieee.org

[2] J. R. Black, "Electromigration—A brief survey and some recent results," *IEEE Trans. Electron Devices*, vol. 16, no. 4, pp. 338–347, Apr. 1969.

[3] I. A. Blech, "Electromigration in thin aluminum films on titanium nitride," *J. Appl. Phys.*, vol. 47, no. 4, pp. 1203–1208, 1976.

[4] R. L. De Orio, H. Ceric, and S. Selberherr, "Physically based models of electromigration: From black's equation to modern TCAD models," *Microelectron. Rel.*, vol. 50, no. 6, pp. 775–789, 2010.

[5] X. Huang, A. Kteyan, S. X.-D. Tan, and V. Sukharev, "Physics-based electromigration models and full-chip assessment for power grid networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 11, pp. 1848–1861, Nov. 2016.

[6] V. Sukharev, A. Kteyan, and X. Huang, "Postvoiding stress evolution in confined metal lines," *IEEE Trans. Device Mater. Rel.*, vol. 16, no. 1, pp. 50–60, Mar. 2016.

[7] H.-B. Chen, S. X.-D. Tan, X. Huang, T. Kim, and V. Sukharev, "Analytical modeling and characterization of electromigration effects for multibranch interconnect trees," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 11, pp. 1811–1824, Nov. 2016.

[8] V. Mishra and S. S. Sapatnekar, "Predicting electromigration mortality under temperature and product lifetime specifications," in *Proc. Des. Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[9] H.-B. Chen, S. X.-D. Tan, J. Peng, T. Kim, and J. Chen, "Analytical modeling of electromigration failure for VLSI interconnect tree considering temperature and segment length effects," *IEEE Trans. Device Mater. Rel.*, vol. 17, no. 4, pp. 653–666, Dec. 2017.

[10] S. Chatterjee, V. Sukharev, and F. N. Najm, "Power grid electromigration checking using physics-based models," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1317–1330, Jul. 2018.

[11] C. Cook, Z. Sun, E. Demircan, M. D. Shroff, and S. X.-D. Tan, "Fast electromigration stress evolution analysis for interconnect trees using Krylov subspace method," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 5, pp. 969–980, May 2018.

[12] S. Wang, Z. Sun, Y. Cheng, S. X.-D. Tan, and M. B. Tahoori, "Leveraging recovery effect to reduce electromigration degradation in power/ground TSV," in *Proc. Int. Conf. Comput.-Aided Des. (ICCAD)*, Nov. 2017, pp. 811–818.

[13] H. Zhao and S. X.-D. Tan, "Postvoiding FEM analysis for electromigration failure characterization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 11, pp. 2483–2493, Nov. 2018.

[14] L. Chen, S. X.-D. Tan, Z. Sun, S. Peng, M. Tang, and J. Mao, "Fast analytic electromigration analysis for general multisegment interconnect wires," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 421–432, Feb. 2020.

[15] S. X.-D. Tan, M. Tahoori, T. Kim, S. Wang, Z. Sun, and S. Kiamehr, *VLSI Systems Long-Term Reliability—Modeling, Simulation and Optimization*. Cham, Switzerland: Springer Publ., 2019.

[16] M. A. Korhonen, P. Borgesen, K. N. Tu, and C.-Y. Li, "Stress evolution due to electromigration in confined metal lines," *J. Appl. Phys.*, vol. 73, no. 8, pp. 3790–3799, 1993.

[17] V. Sukharev and F. N. Najm, "Electromigration check: Where the design and reliability methodologies meet," *IEEE Trans. Device Mater. Rel.*, vol. 18, no. 4, pp. 498–507, Dec. 2018.

[18] X. Wang, Y. Yan, J. He, S. X.-D. Tan, C. Cook, and S. Yang, "Fast physics-based electromigration analysis for multi-branch interconnect trees," in *Proc. Int. Conf. Comput.-Aided Des. (ICCAD)*, Nov. 2017, pp. 169–176.

[19] M. A. Al Shohel, V. A. Chhabria, N. Evmorfopoulos, and S. S. Sapatnekar, "Analytical modeling of transient electromigration stress based on boundary reflections," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2021, pp. 1–8.

[20] X. Wang, H. Wang, J. He, S. X.-D. Tan, Y. Cai, and S. Yang, "Physics-based electromigration modeling and assessment for multi-segment interconnects in power grid networks," in *Proc. Des. Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2017, pp. 1727–1732.

[21] Z. Long, Y. Lu, and B. Dong, "PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network," *J. Comput. Phys.*, vol. 399, Dec. 2019, Art. no. 108925.

[22] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nat. Rev. Phys.*, vol. 3, no. 6, pp. 422–440, 2021.

[23] C. Cook, Z. Sun, T. Kim, and S. X.-D. Tan, "Finite difference method for electromigration analysis of multi-branch interconnects," in *Proc. Int. Conf. Synth. Model. Anal. Simul. Methods Appl. Circuit Des. (SMACD)*, Jun. 2016, pp. 1–4.

[24] "The OpenROAD project." 2023. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD

[25] M. A. Korhonen, P. Borgesen, D. D. Brown, and C.-Y. Li, "Microstructure based statistical model of electromigration damage in confined line metallizations in the presence of thermally induced stresses," *J. Appl. Phys.*, vol. 74, no. 8, pp. 4995–5004, 1993.

[26] Z. Sun, S. Yu, H. Zhou, Y. Liu, and S. X.-D. Tan, "EMSpice: Physics-based electromigration check using coupled electronic and stress simulation," *IEEE Trans. Device Mater. Rel.*, vol. 20, no. 2, pp. 376–389, Jun. 2020.

[27] W. Jin, S. Sadiqbatcha, Z. Sun, H. Zhou, and S. X.-D. Tan, "EM-GAN: Data-driven fast stress analysis for multi-segment interconnects," in *Proc. IEEE Int. Conf. Comput. Des. (ICCD)*, Oct. 2020, pp. 296–303.

[28] W. Jin, L. Chen, S. Sadiqbatcha, S. Peng, and S. X.-D. Tan, "EMGraph: Fast learning-based electromigration analysis for multi-segment interconnect using graph convolution networks," in *Proc. 58th ACM/IEEE Des. Autom. Conf. (DAC)*, 2021, pp. 919–924.

[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.

[30] W. Jin, S. Peng, and S. X.-D. Tan, "Data-driven electrostatics analysis based on physics-constrained deep learning," in *Proc. Des. Autom. Test Eur. Conf. (DATE)*, Feb. 2021, pp. 1–6.

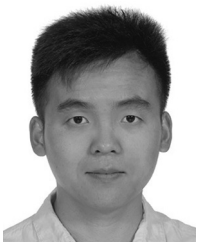
[31] T. Hou, N. Wong, Q. Chen, Z. Ji, and H.-B. Chen, "A space–time neural network for analysis of stress evolution under DC current stressing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5501–5514, Dec. 2022.

[32] V. Dwivedi and B. Srinivasan, "Physics informed extreme learning machine (PIELM)—A rapid method for the numerical solution of partial differential equations," *Neurocomputing*, vol. 391, pp. 96–118, May 2020.

[33] S. Dong and Z. Li, "Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations," *Comput. Methods Appl. Mech. Eng.*, vol. 387, Dec. 2021, Art. no. 114129.

[34] H. Sun, M. Hou, Y. Yang, T. Zhang, F. Weng, and F. Han, "Solving partial differential equation based on Bernstein neural network and extreme learning machine algorithm," *Neural Process. Lett.*, vol. 50, no. 2, pp. 1153–1172, 2019.

[35] E. Galaris, G. Fabiani, F. Calabrò, D. di Serafino, and C. Siettos, "Numerical solution of stiff ODEs with physics-informed RPNNs," 2021, *arXiv:2108.01584*.
[36] S. Rout, V. Dwivedi, and B. Srinivasan, "Numerical approximation in CFD problems using physics informed machine learning," 2021, *arXiv:2111.02987*.
[37] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006, pp. 138–139.
[38] M. Tancik et al., "Fourier features let networks learn high frequency functions in low dimensional domains," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2020, pp. 7537–7547.
[39] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[40] M. Barulli and D. J. Evans, "Implicit Gauss-Jordan scheme for the solution of linear systems," *Parallel Algorithms Appl.*, vol. 10, nos. 1–2, pp. 145–159, 1996.
[41] S. M. Alam, C. L. Gan, F. L. Wei, C. V. Thompson, and D. E. Troxel, "Circuit-level reliability requirements for Cu metallization," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 522–531, Sep. 2005.
[42] L. Doyen, E. Petitprez, P. Waltz, X. Federspiel, L. Arnaud, and Y. Wouters, "Extensive analysis of resistance evolution due to electromigration induced degradation," *J. Appl. Phys.*, vol. 104, no. 12, 2008, Art. no. 123521.
[43] V. A. Chhabria and S. S. Sapatnekar. "PDNSim." 2023. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/PDNSim
[44] Z. Sun, E. Demircan, M. D. Shroff, C. Cook, and S. X.-D. Tan, "Fast electromigration immortality analysis for multisegment copper interconnect wires," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3137–3150, Dec. 2018.
[45] B. Yu et al., "Methodology for standard cell compliance and detailed placement for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 726–739, May 2015.



Liang Chen (Member, IEEE) was born in 1992. He received the B.E. degree in electromagnetic field and wireless technology from Northwestern Polytechnical University, Xi'an, China, in 2015, and the Ph.D. degree in electronic science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2020.

He was a Postdoctoral Researcher with the VLSI System and Computation Laboratory, Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA, USA, from 2020 to 2022. His current research interests include machine learning for VLSI reliability analysis, signal integrity of high-speed interconnects, electrothermal co-simulation of 3-D integrated packages, and electromigration reliability.

Wentian Jin (Student Member, IEEE) received the B.S. and M.S. degrees in instrument science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA, USA.

He is currently a Student Researcher with the VLSI Systems and Computation Laboratory, University of California at Riverside. His current research interests are electronic design automation and applied machine learning in VLSI reliability analysis.

Mohammadamir Kavousi (Graduate Student Member, IEEE) was born in 1990. He received the B.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2016, and the M.Sc. degree in electrical and computer engineering from Southern Illinois University, Carbondale, IL, USA, in 2019. He is currently pursuing the Ph.D. degree with the VLSI System and Computation Laboratory, University of California at Riverside, Riverside, CA, USA.

His current research focuses on machine learning approaches for VLSI reliability analysis, the impact of Joule heating-induced temperature gradients on electromigration, and thermomigration and electromigration modeling and simulation.

Subed Lamichhane (Student Member, IEEE) was born in 1992. He received the B.E. degree in electronics and communication engineering from the Advanced College of Engineering, Tribhuvan University, Kirtipur, Nepal, in 2013, and the M.Sc. degree in electrical engineering from Lamar University, Beaumont, TX, USA, in 2020. He is currently pursuing the Ph.D. degree in electrical engineering with the University of California at Riverside, Riverside, CA, USA.

His research interests include VLSI reliability, VLSI CAD, VLSI EDA, and machine learning.

Sheldon X.-D. Tan (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 1992 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Iowa, Iowa, IA, USA, in 1999.

He is a Professor with the Department of Electrical Engineering, University of California at Riverside, Riverside, CA, USA, and also a Cooperative Faculty Member with the Department of Computer Science and Engineering. He was a Visiting Professor with Kyoto University, Kyoto, Japan, as a JSPS Fellow from December 2017 to January 2018. He has published more than 330 technical papers and has coauthored six books on those areas. His research interests include machine and deep learning for VLSI reliability modeling and optimization at circuit and system levels, machine learning for circuit and thermal simulation, thermal modeling, optimization and dynamic thermal management for many-core processors, efficient hardware for machine learning and AI, and parallel computing and simulation based on GPU and multicore systems.

Dr. Tan received the NSF CAREER Award in 2004, the Best Paper Awards from ICSICT'18, ASICON'17, ICCD'07, and DAC'09, and the Honorable Mention Best Paper Award from SMACD'18. He is serving as the TPC Chair for ASPDAC 2021, and the TPC Vice Chair for ASPDAC 2020. He is serving or served as an Editor-in-Chief for *Integration, the VLSI Journal* (Elsevier), the Associate Editor for four journals: IEEE TRANSACTIONS ON VERY LARGE-SCALE INTEGRATION (VLSI) SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, *Microelectronics Reliability* (Elsevier) and *Electronics* (MDPI), *Microelectronics* and *Optoelectronics* Section.