Machine Learning for Semiconductor Test and Reliability

(Invited Paper)

Hussam Amrouch*, Animesh Basak Chowdhury†, Wentian Jin‡, Ramesh Karri†, Farshad Khorrami†, Prashanth Krishnamurthy†, Ilia Polian*, Victor M. van Santen*, Benjamin Tan†, Sheldon X.-D. Tan‡
* Computer Science Department, University of Stuttgart, Stuttgart, Germany
† Tandon School of Engineering, New York University, New York, USA
‡ University of California, Riverside, California, USA
Email: amrouch@iti.uni-stuttgart.de

***Abstract*—With technology scaling approaching atomic levels, IC test and diagnosis of complex System-on-Chips (SoCs) become overwhelming challenging. In addition, sustaining the reliability of transistors as well as circuits at such extreme feature sizes, for the entire projected lifetime, also become profoundly difficult. This holds even more when it comes to emerging technologies that go beyond convectional CMOS in which the underlying physics are not yet fully understood. In this special session paper, we describe the usage of machine learning in several test and reliability related areas. First, we demonstrate the vital role that machine learning can play in IC test showing the importance of explainability as a frontier for machine learning in IC test. Afterwards, we discuss how novel physics-informed neural networks (PINN) can be employed to model electrostatic problems in VLSI designs. This is essential to mitigate the deleterious effects of electromigration, which is the key source of reliability degradations for interconnects in circuits. Finally, we discuss the major sources of reliability degradations at the transistor level in advanced technology nodes such as transistor aging phenomena and self-heating effects as well as we demonstrate how machine learning approaches can further help in developing reliable emerging technologies.**

***Index Terms*—Machine learning, IC Test, Reliability, Electromigration, BTI, HCI, Emerging Technology**

I. Introduction

Machine learning (ML) is revolutionizing our lives. Problems that were considered intractable a few years back, from generating naturally-sounding speech [1] to detecting fake faces in videos [2], are becoming practical now. While most of the underlying ML concepts had been known since years and decades, ML's recent breakthrough has one major driver: the availability of sufficiently fast and powerful hardware to run sophisticated ML computations. Even before the advent of ML, computing was characterized by sustainable exponential growth enabled by a fundamental positive feedback loop: Advances in electronic devices and design methods lead to better computers, whereas better computers facilitate the development of improved electronic devices and design methods. ML is a new ingredient in this "virtuous cycle" of computing, with the potential of establishing more and stronger positive feedbacks. If using ML for designing computers will result in more computational power becoming available, ML tasks themselves will be the first ones to profit, enabling more advanced types of ML and even better effects on ML-enabled design automation and all the other ML-enabled applications.

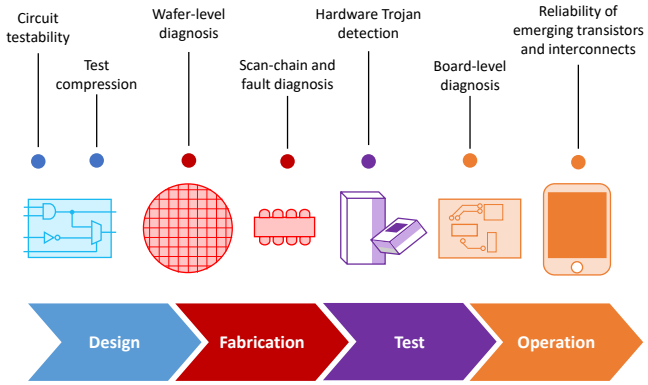


Fig. 1: Design steps of electronic circuits to ensure reliable and secure chips.

Design of electronic circuits is a complex and distributed process broken down in many steps, as seen in Fig. 1. In this paper, we focus on ML in one important sub-field of this process: achieving a circuit's quality, both directly after manufacturing (test) and during its lifetime (reliability). Test and reliability are essential features of today's electronics, which is often intended for use in critical applications, from industry automation to self-driving vehicles. When considering these features, the still-exponentially-growing size and architectural complexity of circuits being designed meets with intricacies of new manufacturing technology nodes. Traditional analytic models for, e.g., reliability prediction start losing their accuracy because second- and third-order effects become prevalent, calling for replacing familiar closed-form equations by trained neural networks. Traditional algorithms for, e.g., fault diagnosis or test point insertion reach their limits for multi-billion-gate designs, and alternative ML-based approaches can provide reasonable alternatives.

II. Explainability as a Frontier for Machine Learning in IC Test

A. Overview

The increasing complexity of IC designs naturally motivates the use of machine learning (ML) techniques for overcoming

scalability issues in IC test [3], [4]. For instance, problems such as test point insertion have been tackled through the use of ML models, obviating costly fault simulations [5], [6]. However, despite apparently promising results, these approaches appear to suffer from potential robustness issues [7] stemming in part from the "black-box" nature of ML models (especially, state-of-the-art deep learning approaches). When using any ML model, a fundamental question often arises: why did an ML model make a given decision? As we move towards integration of ML in-the-loop, there is growing appreciation of the need for explainability and interpretability of models and their decisions/predictions, especially in "mission-critical" applications [8]. *Can explainability approaches benefit the next steps for ML in IC test?* In this section, we will explore explainable ML and pose open areas for future exploration.

B. Explainability Approaches for Machine Learning

Explainability (a term often used interchangeably with *interpretability*) is concerned with the degree to which humans can understand the cause of a decision [9]. The quality of an explanation depends on myriad factors, typically intuited by practitioners in a given domain[1] and is crucial for building *trust* in intelligent, autonomous systems. Explainability makes models more transparent and thus more accountable; insights into otherwise opaque models can assist with identifying biases, enabling sanity checks, and debugging models that fail unexpectedly.

Some ML models are intrinsically explainable to a varying extent, while numerous post hoc techniques can be used to design interpretable approximations of complex models. For example, models such as linear regression and design trees make decisions/predictions in a fairly open way. Humans select and engineer meaningful features from the data upon which the model learns to make predictions; the effect of feature values in an instance (sample) on the overall prediction can be understood by examining the learned weights together with the feature value. Typically, approaches like linear regression oversimplify the decision space which can limit the model's accuracy for complex problems.

Other ML approaches, such as deep learning, are considerably more opaque in what they learn. For example, in a convolutional neural network used for image classification, the inputs are usually raw pixel values; each layer in the neural network, comprising large quantities of artificial neurons with weighted inputs and non-linear activations, transforms the input. Manual inspection to try to link output predictions to individual pixel values is not helpful for understanding why a prediction is made. Essentially, deep learning models learn to extract features without any requirement that the learned features are human-interpretable. To improve explainability, post hoc techniques can be used to produce additional models that provide an explanation. Examples of such techniques include SHAP [12], which predicts the "contribution" of each feature to a given prediction by approximating Shapely values and LIME [13], where local explanations are derived by perturbing an instance of interest, using the "black-box" classifier to get labels, and training a linear classifier on these new samples around the instance of interest.

[1]See the survey by Miller [10] for a comprehensive take on social science and philosophical insights and their intersections with explainable AI.

C. Uses for Explainability

Explainability provides insight into which features, and the (relative) importance of those features, factor most when an ML model makes a decision. In other words, we can use explainability techniques to characterize and scrutinize trained models—we can try to see if models learn things that make sense at a fundamental level. This is readily apparent in simpler ML approaches that are more explainable; for instance, Zhang et al. proposed a single-layer artificial neural network for board-level diagnosis and repair [11] where the neuron weights provide a way to interpret the relative importance of the input features (in this work, the inputs are fault syndromes). An explanation that appears contradictory to a test engineer during model validation can guide inspection of the training data.

Understanding the reasons behind a model's decision can also assist with intervention; in other words, given an explanation, one can be guided towards more appropriate actions. We illustrate this in Fig. 2, where the added explanation can inform subsequent design decisions (either taken by a human-in-the-loop or by autonomous means). There could be different feature values that lead to the same model output, reflecting different root causes which may not be obviously apparent from the data but clearly identified by the ML model. For example, test point insertion works to address "hard-to-test" locations in a design. An ML model can be trained to classify such nodes, given a netlist—explaining whether a node is hard to test because of controllability or observability (or both) can guide the designer towards the type of test point to insert.

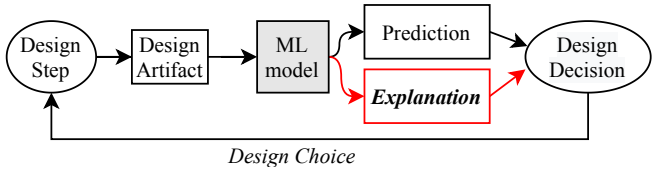


Fig. 2: Explainable ML-in-the-loop can help guide design decisions

In Table I we present a number of example applications in IC test and potential explanations that might be useful, either by providing additional context for a given decision, or providing a way to characterize what a model may have learned. However, the usefulness of explainable ML is determined in part by the nature of the data and the given application—this can also affect the upfront choice of type explainability sought (whether intrinsic or post hoc). In the next subsection, we provide a summary of recent work that explored the use of an intrinsic ML approach.

D. Example: SiFS for Test Point Insertion

As an example of using explainable ML, let us discuss the study by Krishnamurthy et al. [14] and the multi-stage

TABLE I: Potential Uses for Explanations From Machine Learning Models in IC Test Applications

Application	Inputs	Output Predictions	Potential Explainability Contributions
Wafer-level Diagnosis	Image of the wafer + specification test result	Faulty/Clean Die label	Explanations of the factors that contributed to a classification (e.g., relative weight of proximity vs. resemblance to learned pattern(s)) which can offer guidance for subsequent testing
Scan-chain Diagnosis	Failure Vector	Faulty Scan Cell	Explanations as to which output has the most impact on the prediction could guide closer inspection of the corresponding logic cone and scan cells
Fault Diagnosis	Log files	Successful/failed diagnosis, Classify type of failure, large/short runtime	Explanations as to the possible causes for which the model succeeded/failed to generate diagnostic report. For the type of failure, the factors responsible for reporting combinational/sequential logic failure, and presence/absence of intermediate debugging elements' log data responsible to generate faster/longer time to generate diagnostic report.
Board-level Diagnosis	Failure syndrome data from early stage production failed boards	Defective component in recently manufactured boards	Explanations as to which syndrome(s) contributed most to the predicted defective component. (e.g., as in [11])
Test Compression	ATPG log data	Optimal PRNG length	Explanations to see which features of the log data are important and sanity checks that the decision boundary learned is related to the PRNG length
Circuit Testability	SCOAP, COP, Structural features (fanout, type of gates etc)	Optimal test insertion location	Explanations to see which feature(s) are used by the model to make predictions as well as sanity checks that the models are learning
Hardware Trojan Detection	Gate-level netlist, (Random/Directed) Simulation Results	Suspected Hardware Trojans	Explanations for suspicion: structural similarities to training samples or mostly because of simulation inputs/outputs

reasoning process that is entailed in training an intrinsically interpretable model for "human-readable" explanations. This work applied a technique called Sentences in Feature Subsets (SiFS) [15] to node classification for test point insertion, with the objective to predict nodes in a netlist as hard-to-test or otherwise, as illustrated in Fig. 3. While the desire for high accuracy is a given in any application, motivations for explainable ML in this case study included the desire to address the following:

- Which features in the data are relevant for classification? Designers have intuitions as to several features that *could* help with the target objective. In this example, including structural features like (logic-level) distances to (from) primary outputs (inputs), as well as SCOAP values in a node's neighborhood).
- What combinations of features (i.e., combinations of inequalities involving subsets of features) can be used to classify? While there can be several combinations that correlate well with the desired predictions, the exact combination and relative importance of the features is non-obvious up front.
- Given an input data point, what is the best context-sensitive explanation for classification of that data point? Once trained, experts can easily scrutinize the trained model's decision-making for a given input, as well as across inputs, potentially revealing insights into what has been learned. This can be especially useful, particularly in light of suggestions for considering robustness issues in ML for IC test [7].

SiFS produces a classification model as an `OR`-combination of a set of "sentences," where each sentence is defined as an `AND`-combination of "words"—each word is an inequality involving a subset of features. As explained by Krishnamurthy et al. [14],

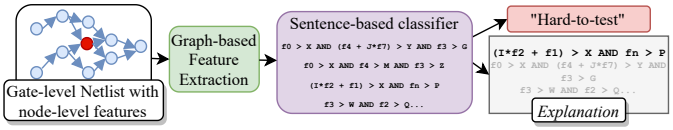


Fig. 3: The SiFS-based classification flow for identifying hard-to-test nodes [14]. After the training phase, only the features that appear in the "sentences" need to be extracted from unseen designs. The ML model produces both a predicted label as well as a context-sensitive, "human-readable" explanation.

each sentence acts as a classification rule, estimating a sub-volume of an overall volume in feature space where training samples of a given class are situated. If any of the learned sentences (rules) are satisfied by a new sample, that sentence is considered "triggered". As a single sample can trigger several sentences simultaneously, the "best" sentence can be picked by selecting the sentence with the highest activation value; this sentence provides a "human-readable" explanation for a given decision as a subset of features and combinations of features as a Boolean combination of inequalities.

Thus, to decide which features are most relevant, the overall architecture is multi-step: feature down selection is first performed by training random forests with random feature subsets and iteratively mutating/evolving the feature subsets using genetic algorithm-based search. Random forests, themselves intrinsically explainable to some degree, allow one to estimate the utility of a feature. These most relevant features are then used in the sentence-based classifier training process, where the "goodness" of a sentence is in part influenced by its complexity (taken as a weighted sum of the effective word lengths in the sentence and the number of distinct features in it). This captures the intuition that should be more compact

for better "readability" of an explanation (viz. Occam's razor).

As an example of the type of sentences (explanations) involved in a SiFS-based classification, consider the example illustrated in Fig. 4. This sentence was picked as best for the majority of test data points correctly classified as hard-to-test for stuck-at-0 faults. The overall classifier for these types of nodes was reported to have an F1 score of 0.942 (see §4.3 and §4.4 of reference [14]). Recall that sentences comprise words in the form of inequalities of feature values; in this example, the SiFS model learned to recognize SCOAP observability of the node, observability of predecessor nodes, paths to primary outputs, and SCOAP controllability-1 as relevant features for the classification task.

How is this explanation useful? Each "word" captures a necessary condition, learned by the ML model, for classifying a sample into a given class. In this example, a test engineer can inspect the sentence and judge whether it is consistent or contradictory to the engineer's intuitions by interpreting the presence/combinations of features in each word, the relative weights for each feature, and the overall sentence. In this example, the feature values are always positive.

In Fig. 4, word 1 suggests that hat the classifier has learned that nodes connected to a high number of primary outputs but with local neighbors (predecessors) with low observability could be hard-to-test—f3 and f2 have positive weights that indicate that increasing feature value (decreasing observability) have more influence on testability than the observability of the node's itself (f1). Word 2 adds another condition that a node that is itself hard to control is likely to be hard-to-test, but only if its neighbors also have low observability—note how f4 has a large negative weight relative to the other weights. If controllability of the node is poor, f4 is high, which means that the local and neighborhood observability also needs to be poor (f1 and f2 high) for a node to be considered hard-to-test. Word 3 has the same feature subset as word 2, but instead adds further insight that nodes that have poor observability (high f1) are only hard-to-test if they also have poor controllability or are in a neighborhood with poor observability, with the neighborhood's observability being more important (given the larger weight for f2). All of these conditions need to be satisfied for this specific sentence to be "triggered" (resulting in a hard-to-test classification), while other conditions are present in other learned sentences, each of which can be similarly analyzed.

E. Discussion and Outlook

The SiFS example illustrates the benefits of intrinsically explainable ML models; in this case, for both local explanations (one can interpret the sentences triggered by a sample) as well as a way to inspect the model as a whole (by examining the family of sentences). One can gain further insights by considering how often a sentence is most strongly activated by samples in a given class, which also aids in characterizing the test data. We turn our attention now to some challenges and open areas in ML for IC test for future exploration.

Where is explainability appropriate? The answer to this question is highly dependent on the application objectives;

```
word 1: (-0.011*f1+0.53*f2+1*f3 > -0.38)
word 2: AND (0.98*f1+0.32*f2+1*f3-2.5*f4 >
-0.091)
word 3: AND (-0.11*f1+1*f2+0.039*f3+0.33*f4
> -0.052)
```

Fig. 4: An example sentence (explanation) for classifying a node as hard-to-test (stuck-at-0) from [14]. f1 is observability, f2 is the average observability among predecessors, f3 is the percentage of output pins where there is more than one path from the node-of-interest to the pin, f4 is controllability-1 of the node itself.

thus an open area to explore is the identification of which IC test applications can benefit from explainability. As observed by Weld and Bansal [8], low-stakes decision-making might not require the additional effort; however, as engineers seek to apply more complex end-to-end ML models, the need for explainability can grow. As Chowdhury et al. observe [7], robustness is often a shortcoming in such ML models. Hence, post hoc explainability techniques, like LIME [13] can provide insights into local decisions, especially when models fail. Analyses with such tools can give greater confidence into ML-based approaches by peering into fundamental concepts learned by the model.

How "much" explainability is needed? As we saw with the SiFS example, designers first intuit features that might be useful for the ML model to achieve its objective; they will choose feature(s) that they think (or know) has some correlation to the desired predictions. This, in turn, can lead to more immediately meaningful explanations from intrinsically explainable models, as well as allows experts to check that their intuitions are satisfied (or probe deeper if they are contradicted). This can build trust in the model. However, intrinsically explainable ML approaches are often associated with poorer accuracy compared to deep learning approaches [8]—sometimes application necessitate more complex approaches with high dimensional data. In fact, it could be the case that ML could learn "meaningful" concepts that go beyond human comprehension. Thus, exploring the continuum of comprehensibility/fidelity of an explanation, alongside performance, is another open area to explore. For example, it can be possible to combine feature representations learned by deep learning models with rule-based ML approaches (like SiFS), producing hybrid models which still produce explanations.

III. Physics-Informed Convolutional Neural Network for Electrostatics Analysis

In this section, we discuss a novel data-driven mesh-less 2D numerical technique for electrostatic analysis using physics-informed convolution neural networks. Physics-informed neural networks (PINN) basically translate the solving of differential equation into an DNN-powered optimization problem with loss functions to incorporate the physics-laws. The solution is obtained when all physics-laws and boundary conditions are constrained or complied after the unsupervised learning. In this work, we exploit the convolution neural networks

(CNN) based PINN as CNN is more amicable for 2D VLSI layout image inputs. We show that the numerical differential operations can be obtained by means of convolution filtering. Numerical results demonstrate that our model achieves an error rate of 9.3% in electric potential estimation without labeled data and yields 5.7% error with the assistance of limited number of coarse labeling data.

A. Background

Electrostatics is an important subject of study as it is pivotal in many VLSI modeling applications such as DC current or power analysis of power and ground networks at the circuit and board levels, capacitance parasitic extraction etc. The goal is to compute voltage potential and electric fields with some voltage and current boundary conditions for dielectrics and metal interconnects or planes.

Traditionally, this differential equation problem is mainly solved by numerical methods such as finite difference or finite elements in which a mesh will be generated to discretize the governing equation, which can be very computationally expensive, especially for large problems. Recently, physics-informed neural networks (PINN) or physics-constrained neural networks (PCNN) have been proposed to learn and encode physics laws expressed by nonlinear partial differential equations (PDE) for complex physical, biological or engineering systems [16], [17]. However, only simple problem has been demonstrated so far [17]–[21]. Recently PINN/PCNN concept has been applied to solve electrostatic problem for VLSI layout [22]. However, this work still use traditional multi-layer perceptron (MLP) for underlying DNN network, which may not be friendly to the 2D VLSI layout inputs.

In this work, we exploit the convolution neural networks (CNN) based PINN as CNN is more amicable for 2D VLSI layout image inputs. We show that the numerical differential operations can be obtained by means of convolution filtering. The preliminary results demonstrate that the proposed physics-informed CNN network can get accurate results around the boundary of the problems. The new solver achieves an error rate of 9.3% in electric potential estimation without labeled data. To further improve the accuracy, some easily accessible coarse labels are introduced at some collocation points derived from the FEM analysis. The resulting solver yields 5.7% error with the assistance of those coarse labeling data.

1) Electrostatics problem: As mentioned above, many VLSI related problems can be concluded to electrostatics problem, where electric current does not exist and there are only static electric fields due to the voltages applied or charges. They are governed by the first equation of the Maxwell's equations, also known as Gauss's law:

$$\nabla^2 u(x) = \frac{-\rho}{\epsilon}, x \in \Phi \tag{1}$$

with following Dirichlet and Neumann boundary conditions:

$$\begin{aligned} u &= f(x),\ x \in \Gamma_D, \\ \nabla u \cdot \vec{n} &= g(x),\ x \in \Gamma_N, \end{aligned} \tag{2}$$

where Φ is the solution domain, Γ_D is the part of the boundary where Dirichlet (voltage) boundary conditions are given, Γ_N is the part of the boundary where Neumann boundary conditions are given, $u(x)$ is unknown potential to be found, ρ is the charge density, ϵ is permittivity, $f(x)$, and $g(x)$ are given voltage sources and current sources at the boundaries.

In cases where static charges are absent, which this work focuses on, 1 becomes Laplace equation:

$$\nabla^2 u = 0 \tag{3}$$

After solving eq. 3, distribution of electric field is usually obtained by calculating the gradient as per its definition:

$$\vec{E} = -\nabla u \tag{4}$$

Solving for $\vec{E}$ under given voltage boundary conditions $f(u)$ is often of more interest for many practical problems. For example, for capacitance extraction, first the voltage is set to $1V$ (VDD) for one interconnect wire (indexed i), $0V$ (GND) for other wires. Then the induced static charge in any other wire j can be computed using Gauss's flux theorem.

2) Finite element method: In convention, electrostatics problems are solved using discretization methods such as FEM or FDM. Results given by commercial tools based on FEM such as COMSOL are usually deemed golden.

In conclusive words, FEM first discretizes the domain to be solved by a mesh. The mesh and the shape function chosen define a function space. Elements of the function space are defined by expansion coefficients of the shape functions. A numeric solution to the original PDEs can be then found by searching in this function space for the one that best fits the original equations. This is done by setting up and solving a linear system from the original PDEs, with the expansion coefficients to be solved. Typically, the final linear system is composed of tens of thousands of unknowns, known as DOF (degree of freedom). Solving a problem of this scale is not very expensive, yet is still noticeable in a longer routine.

B. The proposed physics-informed convolutional neural network solver for electrostatics

1) Label-free PINN model for electrostatics analysis: PINN essentially leverage the well-known capability of DNN as universal function approximation [16], [17]. PINN learns to model the behaviors of any dynamic time-dependent, nonlinear system, expressed by the given PDE with boundary and initial conditions. For electrostatic problem as we see from 1, we are computing the steady-state solution and a typical layout and its electric potential is shown in Fig. 6.

In this work, similar to problems in computer vision, we treat the input-output map as an image-to-image problem in which the input is the 2D layout (Fig. 6a) and the output is the electric potential map (Fig. 6b). As shown in Fig. 5, a CNN-based encoder-decoder network is employed as the backbone of our proposed model. It is composed of two parts, the first half is the encoder consisting of 6 convolutional (Conv) layers, and the second half is the decoder consisting of 6 deconvolutional (Deconv) layers ending with an extra Conv layer. This extra Conv layer is added to smooth the electric potential result and eliminate the checkerboard artifacts.

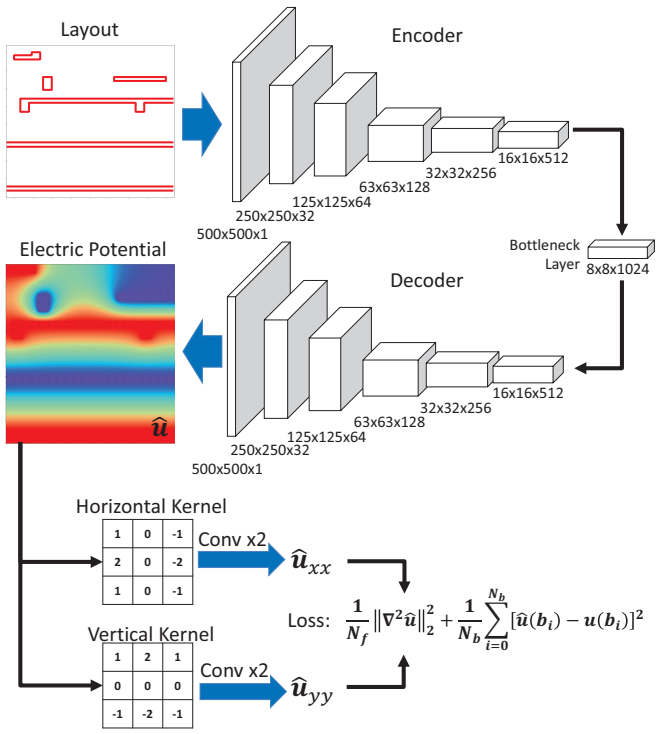


Fig. 5: Architecture of the proposed physics-informed convolutional neural network

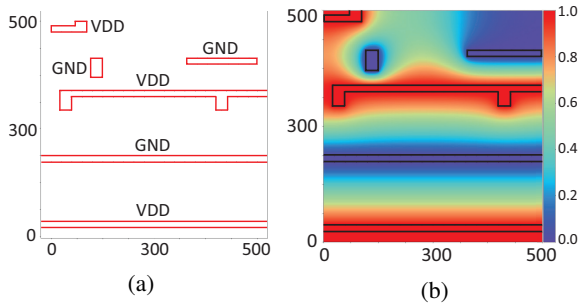


Fig. 6: (a) VLSI interconnects layout (b) Ground truth electric potential

In contrast to traditional data-driven methods, we propose to employ the governing Laplace equation instead of bulky training dataset to train the model, which significantly reduces the training cost and yields much better generalizability. To get the required items in loss function (7), we apply two convolutions with fixed kernels onto $\widehat{u}$, which is the electric potential estimated by the encoder-decoder network. We use two Sobel kernels here, one horizontal and one vertical, which can efficiently obtain the spatial gradients. The resulting $\widehat{u}_{xx}$ and $\widehat{u}_{yy}$ are then employed into the calculation of the PDE loss part in the loss function.

The training process of PINN is essentially an optimization process, it looks for a set of parameters $(\mathbf{W}, \mathbf{b})$ which minimizes the physics-based loss defined by the original differential equation.

Specifically for our electrostatics problems, the physics based loss function can be defined by the original equations (3) and (2)

$$L_{phy}(\mathbf{W}, \mathbf{b}) = \underbrace{||\nabla^2 u||_{\Phi}}_{\text{Gauss's law}} + \underbrace{||u - f(x)||_{\Gamma_D}}_{\text{Boundary condition}} \tag{5}$$

where $||\cdot||$ is L2 norm over a specific domain. Then, training of the network is defined as an optimization problem, looking for the optimal weights and biases $(\mathbf{W}^*, \mathbf{b}^*)$ that minimizes the loss:

$$\mathbf{W}^*, \mathbf{b}^* = arg\, min_{\mathbf{W},\mathbf{b}} L_{phy}(\mathbf{W}, \mathbf{b}) \tag{6}$$

In practice, the L2 norm is computed using the collocation method [23]. The domain Φ and the boundary Γ is discretized into sets of collocation points Φ_d and Γ_d, with the number of points $|\Phi_d| = N_f$ and $|\Gamma_d| = N_b$ respectively. Then the loss is computed through the mean square error (MSE) on these points. For the part that corresponds to Gauss's law in the PDE form,

$$L_{pde}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_f} \left\| \nabla^2 \widehat{u} \right\|_2^2 \tag{7}$$

where $\widehat{u}$ stands for the electric potential estimated by the model, $\nabla^2 \widehat{u} = \widehat{u}_{xx} + \widehat{u}_{yy}$, $\widehat{u}_{xx}$ and $\widehat{u}_{yy}$ are two gradient images along the horizontal and vertical directions estimated by Sobel filter.

For the part that covers the boundary condition,

$$L_{bou}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_b} \sum_{i=0}^{N_b} \left[\widehat{u}\left(b_i\right) - u\left(b_i\right) \right]^2 \tag{8}$$

where $\{b_i\}_{i=1}^{N_b}$ are the collocation points on the boundary, $u(b_i)$ are the ground truth boundary voltages.

Combining the two parts of the loss function, the complete loss function used in practice is defined as

$$L_{pinn}(\mathbf{W}, \mathbf{b}) = L_{pde}(\mathbf{W}, \mathbf{b}) + L_{bou}(\mathbf{W}, \mathbf{b}) \tag{9}$$

By minimizing the loss function, the model output $\widehat{u}$ will converge to an accurate solution to the original problem.

2) Improved loss function with labels: Our study shows that for many practical problems with complicated boundary conditions, PINN loss function defined in (9) may still lead to large errors especially for the region far away from the boundary. In this case, introducing some data from numerical solutions or measurement as labels can be instrumental for the training of PINN networks. For example, one can get the solution with FEM on a coarse mesh, and use it to aid training the model. Compared with the much longer runtime required to solve the PDE on a finer mesh or training the label-free model, the cost of getting such coarse labels is negligible.

Denote the number of coarse labels as N_l, a new part of label-defined loss is then

$$L_{bou}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_l} \sum_{i=0}^{N_l} \left[\widehat{u}\left(l_i\right) - u\left(l_i\right) \right]^2 \tag{10}$$

where $\{l_i\}_{i=1}^{N_l}$ are the collocation points of the coarse labels, $u(l_i)$ are the ground truth voltages at these points.

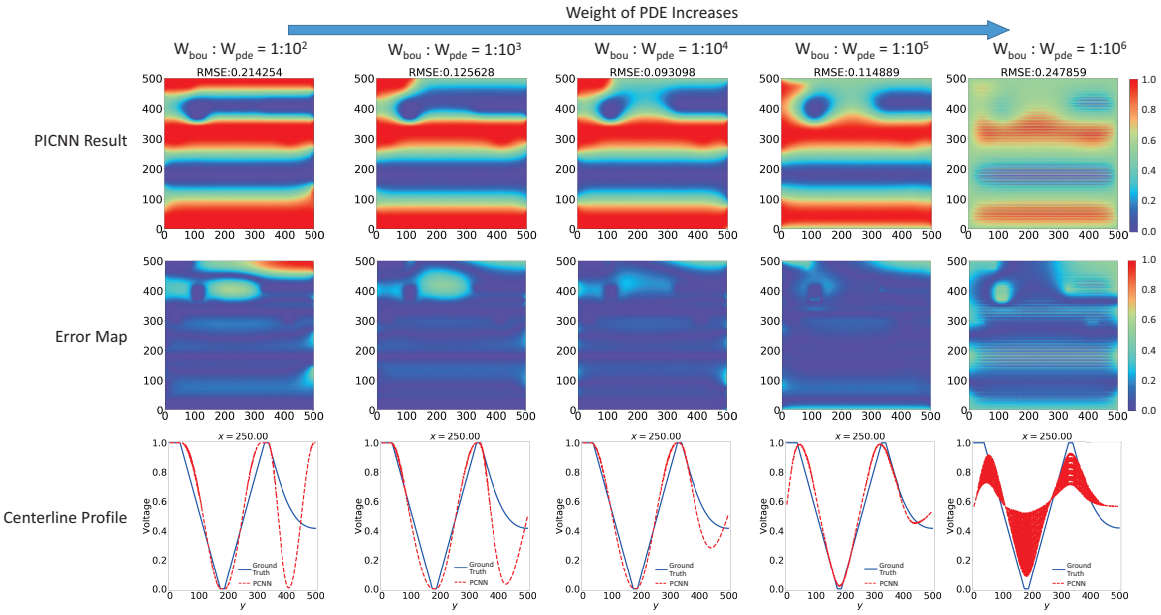


Fig. 7: Influence of weights on PINN result

In practice, we can see these coarse labels as scattered boundary points so that the label data loss can be combined into the boundary loss. Thus, the model is still trained using the loss function shown in (9) but with colocation points of coarse labels appended to the boundary points.

C. Numerical results and discussions

In this section, we present the experimental results of our proposed model in performing electrostatics analysis, including both electric potential and electric field estimations for VLSI interconnects. The model is implemented in Python based on PyTorch(1.6.0) and runs on a single NVIDIA GeForce TITAN RTX GPU. We test our model on a layout extracted from real design synthesized with a 32nm educational technology. Fig. 6 shows the testing layout and its ground truth electric potential.

1) Label-free PINN network: As the loss function in our proposed model consists of two independent parts, i.e. PDE loss and boundary loss, the relative ratio of the weights for each part ($\boldsymbol{W_{pde}}$ and $\boldsymbol{W_{bou}}$) has significant influence on final results. To find the best configuration, we explored different weight ratios as shown in Fig. 7.

When $\boldsymbol{W_{pde}}$ is relatively small, the model focuses more on the boundaries while ignoring transition areas, especially the points that are far from any adjacent boundary. Gradually increasing $\boldsymbol{W_{pde}}$ leads to more smooth transitions between VDD and GND, but may also lose some accuracy on the boundaries. The optimized tradeoff is achieved when $\boldsymbol{W_{bou}}$:$\boldsymbol{W_{pde}}$ is set to 1:10^4, which yields smooth transitions while maintaining acceptable accuracy on boundaries as well.

We compared the estimated electric potential given by our model against the ground truth, the root-mean-square error (RMSE) is 93mV, which is 9.3% in terms of normalized RMSE (NRMSE) when considering the full scale between VDD and GND (1V). Such result is achieved with 1000 epochs of training in 37.6 seconds, which is slightly slower than the 23.14s taken by the COMSOL. However, our model is solving the problem in euclidean space which requires no prior work of meshing, while this is not the case in FEM method. Thus, our model saves much computational cost against FEM method with comparable results. Moreover, the machine learning-based model has greater potential in solving multiple tiles of layouts in parallel which may lead to even larger advantages in terms of computational cost when dealing with large designs.

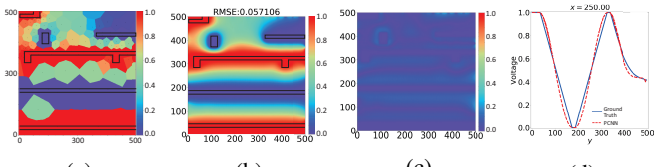
Although the estimated electric potential loses some accuracy, it is still acceptable especially when considering that such result is achieved when the training is purely unsupervised with no labeled data required.

2) Simulation-label assisted PINN: The result achieved by our model with pure unsupervised learning in Section III-C1 is acceptable but still has a large room for improvement. In this section, we propose to take one step back and add some easily accessible labeled data into training to improve the overall accuracy.

As stated in Section III-B1, the reason that we prefer an unsupervised model over a supervised one is mainly because of the great cost it takes to generate a usable training dataset. This cost is particularly large in this application as it usually takes several minutes to mesh and solve a single layout. However, if we just coarsely mesh the layout with less than a hundred grids, the whole process of solving such a low-precision layout can be done with much lower cost in COMSOL. This observation leads to our proposal to employ these easily accessible coarse data to assist the training process of PINN with the hope that the result accuracy can be further improved.

The experiment is conducted using the same layout as Section III-C1 and the added coarse label data points are

shown in Fig. We employed 64 coarse data points which can be simply seen as extra scattered boundary points. Thus, in the training process, these coarse labels are combined into the boundary conditions and enforced by the boundary loss part in the loss function. The label-assisted PINN is trained using the same hyperparameters and runs for 1000 epochs. The estimated electric potential is shown in Fig. 8b.



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

Fig. 8: (a) Coarse label data (b) Label-assisted PINN estimated result (c) Error map (d) Centerline voltage profile

The result accuracy of the label-assisted PINN is significantly improved compared to the unsupervised version. With the assistance of limited number of coarse labels, the RMSE is reduced from 93mV to 57mv, which is 5.7% NRMSE.

IV. Reliability Challenges in Advanced Technologies

Technology scaling improves the efficiency of computing systems. With smaller transistors, computing systems can use more complex logic (more transistors) for more performance at a lower power consumption. However, since geometry scaling of transistors does not follow Dennard scaling anymore, geometry is scaled more than the corresponding supply voltage. Hence, the electric fields rise with each new generation.

This inevitably lead to problems within the transistor, such as insufficient control over leakage. In turn, this leads to innovations in the transistors. With the introduction of 32nm semiconductor technology, high-k dielectrics were introduced and with the introduction of 14nm technology the 3D structure of FinFET transistors was necessary to keep leakage under control.

These ever-increasing electric fields stimulate existing degradation phenomena like aging effects such as Bias Temperature Instability (BTI) and Hot-Carrier Degradation (HCD). Additionally, these transistor innovations (such as FinFET and high-k dielectrics) result in novel reliability challenges (such as the Self-Heating Effect (SHE)) on top of the exacerbated existing ones. In this work, we provide an overview over well-known and newly occurring reliability degradation phenomena. In particular, we discuss how these phenomena can be modeled to study the impact on the chip level, while the defects occur on the transistor level. For this purpose, we discuss various high-performance modeling approaches, abstractions and machine learning techniques.

Furthermore, beyond regular transistor innovations like FinFET, this section discusses emerging technologies such as Negative Capacitance Field-Effect Transistors (NCFET). NCFETs feature a ferroelectric layer in the gate dielectric, which provides voltage amplification and thus better transistor performance at lower supply voltages.

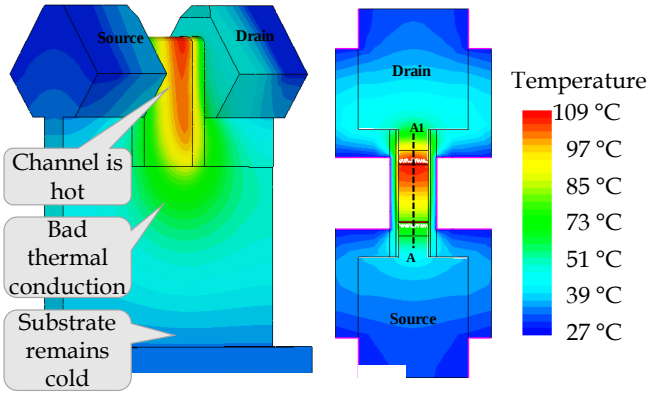


Fig. 9: SHE heating the channel of a 14nm FinFET transistor, reducing channel carrier mobility and thus the ON-current of the transistor. Additionally, aging phenomena are stimulated by these elevated channel temperatures.

A. Self-Heating Effect

The Self-Heating Effect (SHE) is the entrapment of heat within the channel of a transistor. Since the channel of a transistor is just a semi-conductor, significant Joule heating occurs when high currents are flowing through the transistor. In planar MOSFETs, this heat from the channel was dissipated via the substrate to the rest of the chip (where cooling then removes the heat). This heat flux is severely hampered in FinFET transistors, as now the channel is encapsulated on three sides and the connection to the substrate is through a long and narrow fin. Hence, the thermal resistance from the channel to the substrate is much higher in FinFET than planar MOSFET [24].

During the operation of the transistor, the current flowing through the transistor heats the transistor's channel. The high thermal resistance from the channel to the substrate ensures that this heat remains trapped and the channel becomes hot (as shown in Fig. 9). This elevated channel temperature (ΔT_C) has two consequences. First, the ON-current of the transistor reduces, which decreases the performance of the computing system (as clock frequencies have to be lowered) [24]. Secondly, aging phenomena like BTI and HCD are stimulated, since higher temperatures accelerates these physical degradation processes [25].

Table III shows the biggest peaks in the channel temperature distribution during the execution of matrix multiplication in a processor. Since transistors cool and heat up in nano-seconds (minuscule thermal capacitances with large heat fluxes), the temperature varies by over $140°C$ across the processor [26].

With the introduction of the ferroelectric layer in NCFET transistors, self-heating becomes even worse as the thermal resistance of the gate dielectric drops even further. Our work in [27] shows how SHE is even worse in these emerging devides (see Table. II comparing NC-FinFET to regular FinFETs).

B. Transistor Aging

Bias Temperature Instability (BTI) and Hot-Carrier Degradation (HCD) are the key aging phenomena in p- and n-type

Vds	ΔT_C FinFET [$^\circ C$]	ΔT_C NC-FinFET [$^\circ C$]
0.3V	24	28
0.4V	37	44
0.5V	46	55
0.6V	58	70
0.7V	67	79
0.8V	75	88

TABLE II: Additional self heating in NC-FinFET compared to regular FinFET transistors [27].

ΔT_C	% of n-FinFET	% of p-FinFET
0 $^\circ C$	2.5 %	2.4 %
62 $^\circ C$	0.0 %	1.2 %
73 $^\circ C$	0.6 %	0.0 %
121 $^\circ C$	2.7 %	0.2 %
140 $^\circ C$	0.0 %	2.1 %

TABLE III: Overview of the peaks of SHE-induced channel temperature increase across a processor executing a matrix multiplication benchmark. Based on data from [26]

transistor, respectively. The ever-increasing electric fields generate and electrically activate defects within the gate dielectric. These electrically charged defects manifest themselves as a threshold voltage shift in the transistors and lower their ON-current. Hence, these aging phenomena lower the performance of computing systems.

With tiny geometries in the nano-meter range, these defects became countable (tens of defects compared to thousands in 32nm and above) and hence each transistor reacts differently depending its individual defects. This manifests itself as variability through defects [28], shown in Fig. 10. As now the response of each transistor is not deterministic and varies widely, defect-level aging models for BTI and HCD are necessary. Our work in [25] combined BTI, HCD and other degradation phenomena into a single defect-level degradation model and [28] extended that model to include this form of transistor-to-transistor defect-induced variability.

C. Accelerated Reliability Modeling on GPUs

Reliability modeling and simulations are frequently limited in scope and detail (accuracy) by their execution time. It is impossible to simulate each defect in billions of transistors operating at billions operations per second for years of its lifetime. Especially since the reliability models for the phenomena become more complex to model ever-evolving reliability physics on the defect-level for each new technology generation. General purpose computing on graphics cards (GPU) has emerged as a potential solution to this issue. GPUs offer thousands of simple processor cores bound in structure optimized for massively parallel execution of simple tasks. Hence, if the algorithms can be mapped to the simpler execution cores of a GPU (compared to full CPU cores), then parallelism beyond 4096 execution cores is possible in a single graphic card. Our work in [29] showed an implementation of our defect-level aging model presented in [25]. While the original MATLAB implementation of the model took minutes per transistor to estimate its threshold voltage shift (its degradation), our C implementation reduced this to seconds.

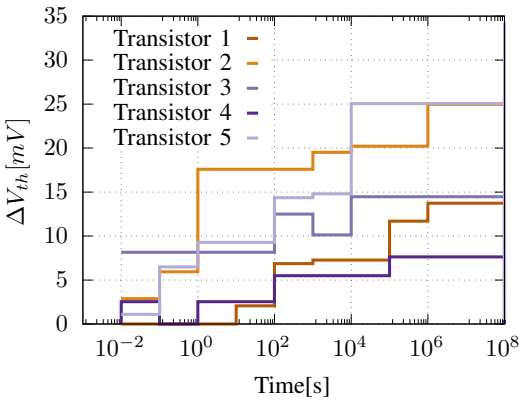


Fig. 10: Variability in the BTI-induced threshold voltage shift, despite the exact same stimuli (same constant voltage and temperature) for each transistor. The differences from transistor to transistor stem from the different defects in their gate dielectrics. This complicates BTI modeling for nano-scale transistors as now to maintain sufficient accuracy, defect-level aging models [25] [28] are mandatory.

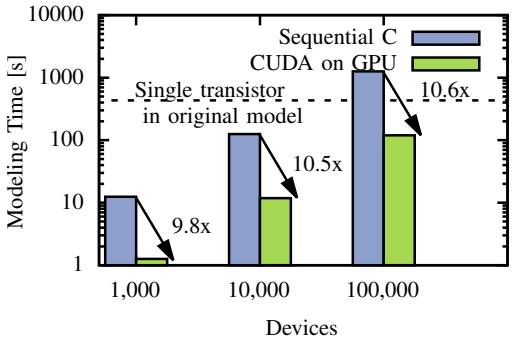


Fig. 11: GPU-based implementation [29] of defect-level aging model from [25] [28]. The GPU implementation can model 100k transistors in under 120 seconds and thus enables delay estimates of large circuits like multipliers.

However, this is not fast enough for larger circuits, whereas our GPU-based implementation can estimate 100k transistors in under 120 seconds. This allows the inclusion of such implementation in automated circuit reliability tools such as CARAT [30], which estimates BTI and HCD on the defect-level for large circuits such as full SRAM arrays (including their sense amplifiers and write drivers) with thousands of transistors.

As circuit reliability estimation tools like CARAT [30] built on top of Analogue Mixed-Signal (AMS) circuit simulators (e.g., SPICE), these tools limit the scope of the reliability estimation. While our work in [29] made the models fast enough for large-scale circuits featuring millions of transistors, the circuit simulators (even with multi-threading enabled) can only handle large circuits on large and expensive compute clusters. Our work in [31] [32] shows how AMS circuit simulator SPICE can be ported to the GPU as well. As Fig.

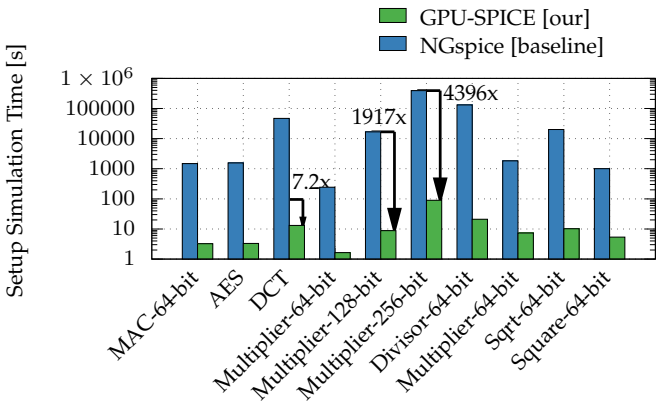


Fig. 12: Reduction in execution time of a phase in the AMS circuit simulator GPU-SPICE [31] [32] comparing the GPU and the multi-core CPU implementations.

12 shows, employing the GPU instead of the CPU allows for much faster execution times. These faster execution times then enable circuit simulations with millions of transistors [32]. If this GPU-SPICE is then used as the backend of CARAT [30], then the impact of BTI and HCD can be estimated fully automatically in large circuits.

D. Machine Learning-based Reliability Modeling

Machine learning (ML) offers suitable solutions to the challenges of reliability modeling. As mentioned in the previous section, execution times are a big challenge for reliability estimations due to their complexity. Additionally, most estimations are performed on sensitive information. The transistor layout, circuit layout (standard cells as well as other overall circuit) and reliability data (aging rates, temperature dependencies, voltage acceleration factors, etc.) are sensitive information for the foundry and design bureau. Since machine learning is intrinsically a black-box approach, it is perfectly suited to abstract and obfuscate the sensitive information, while providing the same functionality (the reliability estimation) to the customer. Additionally, inference in neural network is much quicker than large AMS circuit simulations, i.e. ML can solve both challenges. Our work in [33] showed how the electrical properties and full electrical behavior of transistors in both FinFET and NCFET can be modeled with neural networks. Fig. 13 shows how OFF-current, ON-current, subthreshold slope and threshold voltage can be matched with neural networks (compared to the BSIM-CMG transistor model) in a 14nm FinFET transistor.

However, for large-scale digital circuits, standard cell designs are more typical. Circuits are not build transistor by transistor, but from foundry-provided standard cells, from simple logic gates to complex cells like adders and ALUs. These standard cells can be estimated in AMS circuit simulators (e.g., to find their true worst-case aging for the slow-slow corner [35]) and thus tools like CARAT [30] could be used. However, standard cell characterization involves the simulations of thousands of standard cells under many different signal slews, load

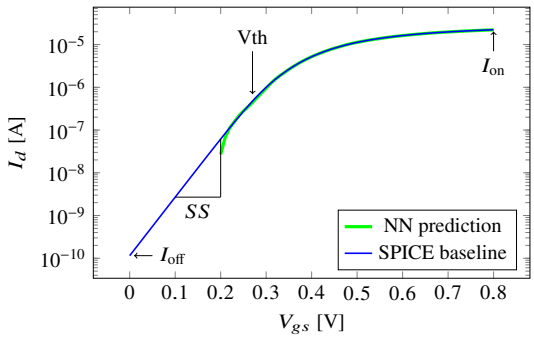


Fig. 13: Neural network transistor model [33] matching the industry-standard AMS transistor model BSIM-CMG in the electrical behavior of a FinFET transistor.

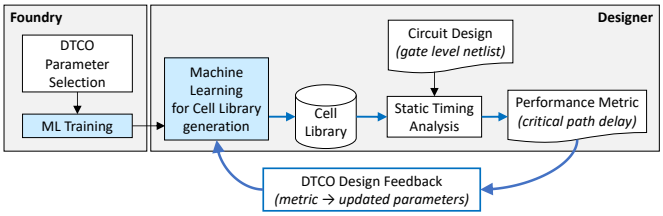


Fig. 14: Design technology co-optimization with our neural-network-based standard cell characterization from [34].

capacitances, temperatures and voltages. Hence, in [34] we provide a machine learning approach to characterize standard cells with neural networks. This approach then enables the reliability estimation of large-scale digital circuits as well as design technology co-optimization (optimizing transistors as well as the circuit itself) as shown in Fig.14.

V. Conclusion and Summary

In this special session paper, we have discussed how machine learning (ML) approaches can play a major role when it comes to IC test and reliability. We summarized the major reliability degradation sources for interconnects and transistors and how advanced neural networks and GPU-based approaches can significantly accelerate reliability estimations which is a key when it comes to analyzing complex SoCs.

Acknowledgement

The work of H. Amrouch, V. M. van Santen, and I. Polian was partially supported by Advantest as part of the Graduate School “Intelligent Methods for Test and Reliability” (GS-IMTR) at the University of Stuttgart.

References

[1] W. Ping, K. Peng, A. Gibiansky, S. Ömer Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, “Deep voice 3: 2000-speaker neural text-to-speech.” *CoRR*, vol. abs/1710.07654, 2017.

[2] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. Canton-Ferrer, “The deepfake detection challenge dataset,” *CoRR*, vol. abs/2006.07397, 2020.

[3] H. Stratigopoulos, “Machine learning applications in IC testing,” in *2018 IEEE 23rd European Test Symposium (ETS)*, May 2018, pp. 1–10, iSSN: 1558-1780.

[4] M. Pradhan and B. B. Bhattacharya, "A survey of digital circuit testing in the light of machine learning," *WIREs Data Mining and Knowledge Discovery*, vol. 11, no. 1, Jan. 2021.

[5] Y. Sun and S. Millican, "Test Point Insertion Using Artificial Neural Networks," in *IEEE Computer Society Symp. on VLSI*, 2019.

[6] 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 *IEEE/ACM Design Automation Conference*, 2019.

[7] A. B. Chowdhury, B. Tan, S. Garg, and R. Karri, "Robust Deep Learning for IC Test Problems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021, accepted.

[8] D. S. Weld and G. Bansal, "The challenge of crafting intelligible intelligence," *Communications of the ACM*, vol. 62, no. 6, pp. 70–79, May 2019.

[9] C. Molnar, *Interpretable Machine Learning*. github.io, 2019, https://christophm.github.io/interpretable-ml-book/.

[10] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, Feb. 2019.

[11] Z. Zhang, K. Chakrabarty, Z. Wang, Z. Wang, and X. Gu, "Smart diagnosis: Efficient board-level diagnosis and repair using artificial neural networks," in *2011 IEEE International Test Conference*, 2011, pp. 1–9.

[12] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 4765–4774.

[13] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, Aug. 2016, pp. 1135–1144.

[14] P. Krishnamurthy, A. B. Chowdhury, B. Tan, F. Khorrami, and R. Karri, "Explaining and Interpreting Machine Learning CAD Decisions: An IC Testing Case Study," in *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*. Virtual Event Iceland: ACM, Nov. 2020, pp. 129–134.

[15] P. Krishnamurthy, A. Sarmadi, and F. Khorrami, "Explainable Classification by Learning Human-Readable Sentences in Feature Subsets," *Information Sciences*, Feb. 2021.

[16] P. P. M. Raissi and G. E. Karniadakis, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," *The Journal of Machine Learning Research*, 2018.

[17] ——, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, 2019.

[18] P. P. M. Raissi and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv e-prints*, 2017.

[19] J. Sirignano and K. Spiliopoulos, "Physics-informed deep generative models," *arXiv e-prints*, 2018.

[20] ——, "DGM: A deep learning algorithm for solving partial differential equations," *Journal of Computational Physics*, 2018.

[21] J. Berg and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," *Neurocomputing*, 2018.

[22] S. P. W. Jin and S. X.-D. Tan, "Data-driven electrostatics analysis based on physics-constrained deep learning," *Proc. Design, Automation and Test In Europe Conf. (DATE)*, 2021.

[23] A. L. I. E. Lagaris and D. I. Fotiadi, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, 1998.

[24] Y. S. C. Om Prakash, Girish Pahwa and H. Amrouch, "Transistor self-heating: The rising challenge for semiconductor testing," in *2021 IEEE VLSI Test Symposium*. IEEE, 2021.

[25] V. M. Van Santen, J. Martin-Martinez, H. Amrouch, M. M. Nafria, and J. Henkel, "Reliability in super-and near-threshold computing: A unified model of rtn, bti, and pv," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 1, pp. 293–306, 2017.

[26] V. M. Van Santen, H. Amrouch, P. Kumari, and J. Henkel, "On the workload dependence of self-heating in finfet circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 10, pp. 1949–1953, 2019.

[27] O. Prakash, G. Pahwa, C. K. Dabhi, Y. S. Chauhan, and H. Amrouch, "Impact of self-heating on negative-capacitance finfet: Device-circuit interaction," *IEEE Transactions on Electron Devices*, 2021.

[28] V. M. van Santen, H. Amrouch, and J. Henkel, "Modeling and mitigating time-dependent variability from the physical level to the circuit level," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 7, pp. 2671–2684, 2019.

[29] V. M. van Santen, J. Diaz-Fortuny, H. Amrouch, J. Martin-Martinez, R. Rodriguez, R. Castro-Lopez, E. Roca, F. V. Fernandez, J. Henkel, and M. Nafria, "Weighted time lag plot defect parameter extraction and gpu-based bti modeling for bti variability," in *2018 IEEE International Reliability Physics Symposium (IRPS)*. IEEE, 2018, pp. P–CR.

[30] V. M. van Santen, S. Thomann, C. Pasupuleti, P. R. Genssler, N. Gangwar, U. Sharma, J. Henkel, S. Mahapatra, and H. Amrouch, "Bti and hcd degradation in a complete $32\times$ 64 bit sram array–including sense amplifiers and write drivers–under processor activity," in *2020 IEEE International Reliability Physics Symposium (IRPS)*. IEEE, 2020, pp. 1–7.

[31] V. M. van Santen, H. Amrouch, and J. Henkel, "Reliability estimations of large circuits in massively-parallel gpu-spice," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 143–146.

[32] V. M. van Santen, F. L. F. Diep, J. Henkel, and H. Amrouch, "Massively parallel circuit setup in gpu-spice," *IEEE Transactions on Computers*, 2020.

[33] F. Klemme, J. Prinz, V. M. van Santen, J. Henkel, and H. Amrouch, "Modeling emerging technologies using machine learning: challenges and opportunities," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–9.

[34] F. Klemme, Y. Chauhan, J. Henkel, and H. Amrouch, "Cell library characterization using machine learning for design technology co-optimization," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[35] V. M. van Santen, H. Amrouch, and J. Henkel, "New worst-case timing for standard cells under aging effects," *IEEE Transactions on Device and Materials Reliability*, vol. 19, no. 1, pp. 149–158, 2019.