Aadam: A Fast, Accurate, and Versatile Aging-Aware Cell Library Delay Model using Feed-Forward Neural Network

Seyed Milad Ebrahimipour[1], Behnam Ghavami[2], Hamid Mousavi[1], Mohsen Raji[3]
Zhenman Fang[2], Lesley Shannon[2]
[1]Shahid Bahonar University of Kerman, {miladebrahimi, hamidmousavi0}@eng.uk.ac.ir
[2]Simon Fraser University, {behnam_ghavami, zhenman, lesley_shannon}@sfu.ca
[3]Shiraz University, raji@shirazu.ac.ir

ABSTRACT

With the CMOS technology scaling, transistor aging has become one major issue affecting circuit reliability and lifetime. There are two major classes of existing studies that model the aging effects in the circuit delay. One is at transistor-level, which is highly accurate but very slow. The other is at gate-level, which is faster but less accurate. Moreover, most prior studies only consider a limited subset or limited value ranges of aging factors.

In this paper, we propose Aadam, a fast, accurate, and versatile aging-aware delay model for generic cell libraries. In Aadam, we first use transistor-level SPICE simulations to accurately characterize the delay degradation of each library cell under a versatile set of aging factors, including both physical parameters (i.e., initial threshold voltage and transistor width/length ratio) and operating conditions (i.e., working temperature, signal probability, input signal slew range, output load capacitance range, and projected lifetime). For each library cell, we then train a feed-forward neural network (FFNN) to learn the relation between the input aging factors and output cell delay degradation. Therefore, for a given input circuit and a given combination of aging factors, we can use the trained FFNNs to quickly and accurately infer the delay degradation for each gate in the circuit. Finally, to effectively estimate the aging-aware lifetime delay of large-scale circuits, we also integrate Aadam into a state-of-the-art static timing analysis tool called OpenTimer. Experimental results demonstrate that Aadam achieves fast estimation of the aging-induced delay with high accuracy close to transistor-level simulation.

KEYWORDS

Aging, Reliability, Delay Model, Machine Learning, Cell Library

ACM Reference Format:
Seyed Milad Ebrahimipour[1], Behnam Ghavami[2], Hamid Mousavi[1], Mohsen Raji[3] and Zhenman Fang[2], Lesley Shannon[2]. 2020. Aadam: A Fast, Accurate, and Versatile Aging-Aware Cell Library Delay Model using Feed-Forward Neural Network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3400302.3415605

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415605

1 INTRODUCTION

With the CMOS technology scaling, the reliability of circuits has become one of the major issues affecting digital circuit designs [1, 7]. In addition to the correct functionality of the circuit, a longtime lifespan is also crucial in many application fields such as aerospace, defense, and medical industries [5, 13]. Transistor aging is a key source of failure that threatens the lifetime reliability of digital circuits. It leads to a degradation of the electrical characteristics of transistors and subsequently, a considerable increase of the device delay [19]. For example, the Negative Bias Temperature Instability (NBTI) aging phenomenon, a major parametric reliability issue, may increase the circuit delay by up to 30% [14]. This impact may eventually lead to violations of circuit timing constraints, reduction of mean time to failure, and faster wear-out of the system.

To steer clear of the aging effects and guarantee the correct functionality of the circuit for the projected lifetime (t), designers have to include a safety timing margin called timing guard band in the design [6, 27]. This timing guard band may decrease the circuit frequency, as shown in Equation 1.

$$frequency = \frac{1}{D\,(t)}\ ;\ D\,(t)\ = D\,(0) + D_{GB}\,(t) \tag{1}$$

where $frequency$ denotes the targeted frequency of the design and $D\,(t)$ represents the lifetime delay. $D\,(0)$ and $D_{GB}\,(t)$ respectively denote the initial delay of the circuit and the delay of the timing guard band. To reduce the performance overhead imposed by $D_{GB}\,(t)$, it is essential to accurately estimate the aging-induced delay degradation of the circuit and apply a minimum $D_{GB}\,(t)$.

However, it is nontrivial to quickly and accurately estimate the aging-induced delay degradation as there are many factors that affect the delay degradation. They include both physical parameters (i.e., initial threshold voltage and transistor width/length ratio) and operating conditions (i.e., working temperature, signal probability, input signal slew range, output load capacitance range, and projected lifetime), which will be explained in detail in Section 2.1.2.

As will be detailed in Section 2.2, the prior studies that model the aging effects in the circuit delay can be divided into the following categories. The first category of work [34] uses full-circuit transistor-level SPICE simulation to achieve high accuracy, but runs at very slow speed. The second category of work [2, 28–30, 37, 40] uses lookup table (LUT) based gate-level models to achieve faster speed at the expense of lower accuracy. The third category of work [16, 26] only performs SPICE simulation for a set of critical paths, but is still slow for large-scale circuits. The last category of work [20, 25, 38, 39] starts to use traditional machine learning techniques such as support vector machine (SVM) and non-linear regression

model to predict the delay caused by the NBTI effect at gate-level or critical path level. In fact, most prior studies only consider a limited subset of aging factors and/or limited value ranges of aging factors.

To address those issues in prior studies, in this paper, we propose Aadam, an accurate yet fast aging-aware delay model for generic cell libraries by considering a versatile set of aging factors that are summarized in Section 2.1.2. The major idea behind Aadam is to leverage 1) the high accuracy of transistor-level SPICE simulation to characterize the delay degradation of library cells only; 2) the learning power of feed-forward neural networks (FFNN) [18] to accurately and quickly predict the aging-induced delay at gate-level under the versatile combinations of aging effects; and 3) the fast aging-aware static timing analysis (STA) of large-scale circuits using the state-of-the-art STA tool called OpenTimer [21].

Aadam provides a fully automated framework for aging-aware STA of large-scale circuits. In Aadam, we first characterize the delay degradation of each library cell under a versatile set of aging factors as mentioned earlier, using the accurate transistor-level SPICE simulations. Based on these characterized data, we train an FFNN to learn the relation between the input aging factors and output delay degradation for each library cell. As a result, these trained FFNNs can quickly and accurately predict the aging-induced delay at gate-level. Finally, we integrate these trained FFNN models to a state-of-the-art STA tool called OpenTimer [21] to effectively estimate the aging-aware lifetime delay of large-scale circuits. We plan to release our Aadam toolflow to the public in the near future.

In our experiments, we use the open-source Nangate 45nm generic cell library [31], and a few circuits from the ISCAS'85 [10], ISCAS'89 [9], ITC'99 [12], and OpenTimer [21] benchmark suites. Compared to the accurate transistor-level SPICE simulation [35, 36], Aadam achieves almost identical accuracy for predicting the lifetime delay of a gate while achieving four orders-of-magnitude speedup. Compared to the prior LUT-based model [2] and SVM-based model [38, 39] that predict the lifetime delay at gate-level, Aadam achieves better accuracy and can accurately predict for input aging factor combinations that are not captured in these models. Compared to the prior critical path based model [20] that performs aging-aware STA at circuit-level, Aadam also achieves better accuracy in predicting the circuit lifetime delay. In addition, Aadam can consistently provide high prediction accuracy when there are perturbations to the circuit, while the prediction accuracy of the critical path based model [20] significantly decreases after the circuit perturbation. Finally, for large-scale circuits with 138.9K to 255.3K gates, our aging-aware delay estimation with Aadam only adds around 5 to 12 seconds extra runtime overhead, which can be well tolerated.

2 RELATED WORK AND OUR NOVELTY

2.1 Aging Effects and Factors

2.1.1 Transistor-Level. The transistor aging phenomenon occurs due to the formation of *interface traps* (breaking of $S_i - H$ bonds at the $S_i - S_iO_2$ interface) and *oxide traps* (capturing of charges in the oxide vacancies within the dielectric). First, during the operation of the transistor, the horizontal electric field over the gate dielectric increases the kinetic energy of the carriers. This results in charge build-up near the transistor drain, leading to defect formation in the drain. These interface traps interact with the charge carriers inside the channel and degrades the transistor's mobility (μ). Second, because of the non-epitaxial structure of S_iO_2, the vertical electric field leads to defect generation in the interface of the transistor and the formation of defects inside the S_iO_2. These defects cause accumulated charges around and within the gate dielectric, which ultimately increases the threshold voltage (V_{th}) of the transistor.

For a projected lifetime (t), the delay of a transistor ($D(t)$) is inversely proportional to its drain current ($I_D(t)$), and $I_D(t)$ is a function of the transistor's mobility ($\mu(t)$) and threshold voltage ($V_{th}(t)$), as shown in Equation 2 (where V_{dd} denotes the supply voltage). As a result, with $\mu(t)$ decreasing and $V_{th}(t)$ increasing during the operation of the transistor, $I_D(t)$ decreases and the transistor delay ($D(t)$) increases.

$$D(t) \propto \frac{1}{I_D(t)}; \qquad I_D(t) \approx \frac{\mu(t)}{2}(V_{dd} - V_{th}(t))^2 \tag{2}$$

2.1.2 Gate-Level. Each gate in the circuit is exposed to different operating conditions that can lead to different aging-induced delay degradations over the *projected lifetime (t)*.

1. The *working temperature (T)* has a great impact on the trap generation of transistors in a gate. As T increases, the rate of interface and oxide trap generation increases [3, 4], i.e., $V_{th}(t)$ increases faster and $\mu(t)$ decreases faster. This includes both Positive Bias Temperature Instability (PBTI) effect in NMOS transistors and Negative Bias Temperature Instability (NBTI) effect in PMOS transistors.
2. The *signal probability (λ)* also has a great impact on the trap generation. The higher the signal probability of the inputs of a gate, the greater the operation cycle (duty cycle) of its transistors. As a result, more interface traps and oxide traps will be generated inside the transistors of the gate, i.e., the aging rate of the gate will increase.
3. The *input signal slew range ($[S_{min}, S_{max}]$)* and *output load capacitance range ($[C_{min}, C_{max}]$)* of a gate can lead to different interface trap generations and different values for $V_{th}(t)$ and $\mu(t)$ of each transistor in a gate [24].
4. Physical parameters, such as *initial threshold voltage ($V_{th}(0)$)* and *transistors' width/length ratio (W/L ratio)*, also affect the aging-induced gate delay.

2.1.3 Circuit-Level. Stemming from the transistor-level, aging can lead to timing violations at the circuit-level. Since each gate of the circuit is operated under a different combination of the above factors, the aging-induced delay degradation of each gate is different. As a consequence, a critical path of the circuit might become non-critical and vice versa over the projected lifetime. Since the lifetime delay of each circuit path is changing due to the aging effects, evaluating aging impacts on the circuit lifetime cannot be determined by only analyzing the critical paths of the circuit. Instead, all of the paths that may violate the timing constraints of the circuit should be jointly considered.

2.2 Aging-Aware Delay Modeling

Although there are a lot of studies that try to model the V_{th} degradation due to aging effects, there are only a few studies that aim to compute the aging-induced gate and/or circuit delay degradation, which can be classified as the following categories.

2.2.1 Full-Circuit Transistor-Level Modeling. An aging-aware transistor-level timing analysis method is introduced in [34]. First, the initial circuit without aging effect is simulated under different operating conditions of each transistor. Then, using the collected information, the V_{th} shift due to aging is computed for every transistor in the entire circuit. Finally, the lifetime V_{th} is applied to each transistor and the circuit delay for the projected lifetime is calculated. Despite the high accuracy, this method is very slow due to transistor-level simulation, which makes it infeasible to be used for large-scale industrial circuits.

2.2.2 LUT-based Gate-Level Modeling. To compensate for the runtime overhead of transistor-level simulations, some studies use gate-level simulation models at the expense of lower accuracy. In [30, 37, 40], a Look-Up Table (LUT) based gate delay model is introduced to correlate the NBTI-induced V_{th} shift of PMOS transistors to the corresponding gate delay degradation. However, it considers neither the PBTI effect in NMOS transistors nor the slope of rising and falling signals of a gate. Kiamehr et. al. [28] propose to use an aging-aware standard cell library. They extend the cell library and provide multiple copies of each library cell considering different input signal probabilities. Then, according to the expected input signal probability of each gate, technology-mapping is done and a robust cell from the extended cell library is used to replace the initial gate. However, they only consider the input signal probability. In [2, 29], an accurate LUT based method is introduced to estimate both NBTI-induced and PBTI-induced delay degradations of each gate. However, all these methods are limited to model the operating conditions under which the LUT is built.

2.2.3 Critical-Path Transistor-Level Modeling. With an eye toward analyzing the effects of aging on complex designs such as processors, Karimi et.al. [26] consider the effects of aging on the critical path only (using the SPICE simulation). However, since the aging effects may change the critical paths into non-critical ones and vice versa, only considering the critical path of the circuit is not sufficient. Another method in [16] considers the effects of the top 20% critical paths. However, this method is infeasible to analyze the aging effects on very large-scale integrated circuits used in the industry. For example, as shown in [14], the b19 benchmark circuit [12] has more than 10^8 paths, and considering the paths with only 5% relative slack time may lead to more than 10^7 paths, which are intractable for analyzing.

2.2.4 Machine Learning based Modeling. Recently, applying machine learning techniques in EDA (Electronic Design Automation) tools has gained increasing attention [22, 23, 32], and machine learning based aging estimation for the circuit is not an exception [20, 25, 38, 39]. In [38, 39], a two-stage workload-dependent NBTI model has been introduced. First, a method is proposed to find a set of representative flip flops that are important to be tracked with regard to the aging effects. And then, they calculate the aging-induced delay degradation of these flip flops by sensing their actual signal probabilities at runtime. They use a Support Vector Machine (SVM) model to map the signal probabilities of these flip flops to the delay degradation of the circuit. However, they assume other parameters such as supply voltage and temperature as constants and only consider the signal probability.

In [25], a non-linear regression model has been introduced to estimate the delay degradation of a circuit due to NBTI effects. This method exploits the critical path set, which consists of the critical path and near-critical paths with delays higher than 80% of the critical path delay. Then, using a non-linear regression model, various NBTI operating conditions are mapped to the delays of the critical path set. An extension of this model is presented in [20] to estimate the lifetime delay of the circuit considering time variant operating conditions. As stated before, due to the large number of paths to be considered, both of these methods are impracticable to analyze the aging effects of the large-scale industrial circuits.

2.2.5 Summary. In summary, prior studies for aging-aware delay modeling usually suffer from at least one of the following issues: low speed, low accuracy, or limited consideration of aging factors.

2.3 Our Novelty

In this paper, we propose a feed-forward neural network (FFNN) based method to model the aging-induced delay degradation, which 1) considers a versatile set of aging factors as summarized in Section 2.1.2, 2) achieves high accuracy that is close to transistor-level modeling, and 3) achieves high speed that is close to the gate-level modeling. Moreover, once our FFNN models (one model per library cell) are trained for a cell library at a specific process technology, they can be used to estimate the lifetime delay of any circuit under different aging effects without the need of retraining. Finally, we achieve fast aging-aware static timing analysis (STA) for large-scale circuits by integrating our model with a state-of-the-art STA tool called OpenTimer [21].

3 DESIGN OF AADAM

The overall design of our Aadam framework is shown in Figure 1, which consists of three major phases. First, we use transistor-level SPICE simulation to accurately characterize the delay degradation of each cell from a generic cell library, under a versatile set of aging factors summarized in Section 2.1.2. Second, using the characterized aging-aware cell delay dataset, we train a feed-forward neural network (FFNN) for each cell in a generic cell library [1] to learn the relation between its input aging factors and output cell delay degradation. Using the trained FFNN models, we can infer the circuit delay degradation caused by a versatile set of aging effects with high accuracy and fast runtime, without the need of model retraining. Finally, to effectively estimate the lifetime delay of a large-scale circuit under specified aging effects, we implement an aging-aware Static Timing Analysis (STA) toolflow by extending a state-of-the-art STA tool called OpenTimer [21] with our trained FFNN models.

3.1 SPICE Characterization of Cell Library

The left part of Figure 1 presents our proposed flow to characterize the aging-induced delay for each library cell using accurate SPICE simulation. First, we characterize the threshold voltage (V_{th}) degradation of a transistor in the technology library using SPICE simulation. Here we consider the transistor characteristics, including

[1] A typical generic cell library has around 100 cells, which is much smaller than the number of gates in a typical industrial circuit.

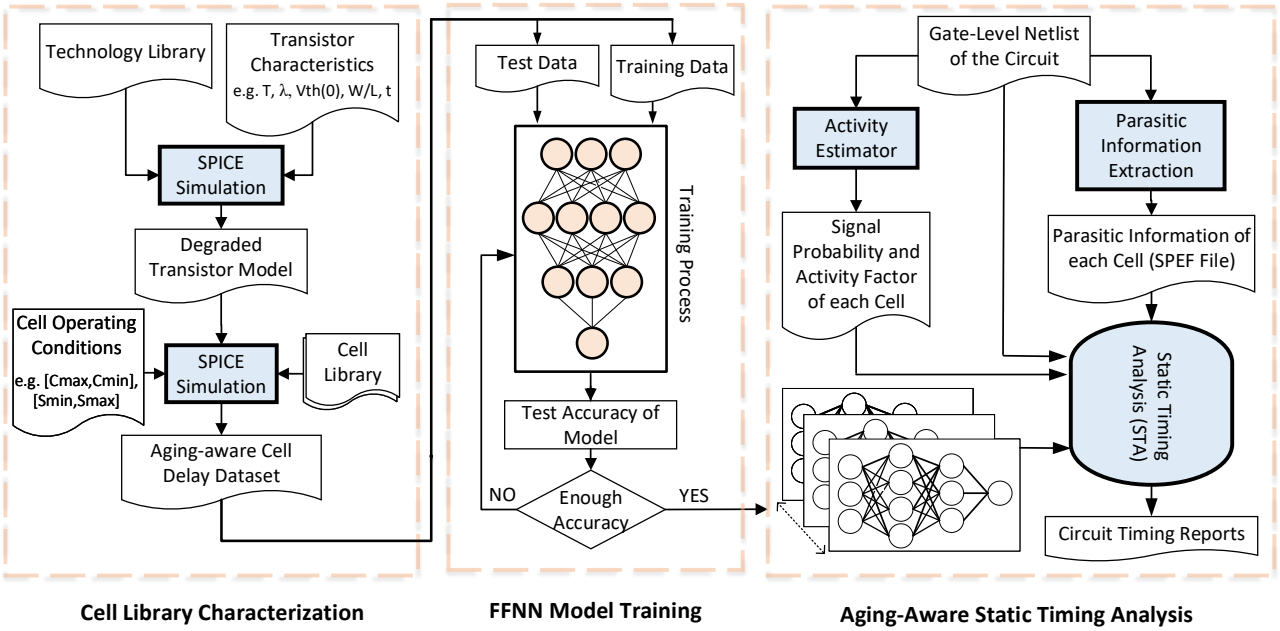


Figure 1: Overview of our aging-aware Static Timing Analysis (STA) framework

the temperature (T), signal probability (λ), initial threshold voltage ($V_{th}(0)$), transistor sizes (W/L *ratios*), and projected lifetime (t) of the transistor. For each combination of these aging factors, a degraded transistor model is created using the SPICE simulation results, including both NMOS and PMOS transistors. Second, for each cell from the cell library, we perform another round of SPICE simulation to calculate the lifetime delay and the slope of the cell. Here we consider the the degraded transistor models for the cell and other cell characteristics, including the input signal slew range ($[S_{min}, S_{max}]$) and output load capacitance range ($[C_{min}, C_{max}]$). Finally, we get an accurate aging-induced delay degradation for each library cell under different combinations of the aging factors.

3.2 FFNN Model for Cell-Level Aging Delay

Influencing the aging-induced delay degradation of a cell from a given combination of its operating conditions and physical parameters (as summarized in Section 2.1.2) can be considered as a regression problem. Therefore, we propose to solve this problem using a feed-forward fully-connected neural network (FFNN) [18].

Figure 2 presents the architecture of our proposed FFNN. The input layer corresponds to the input operating conditions and physical parameters of a gate. The input feature vector X that feed to the input layer can be defined as:

$$\begin{aligned} X &= [x_1, x_2, \ldots, x_d] \\ &= [T, \lambda, S_{min}, S_{max}, C_{min}, C_{max}, W/L, V_{th}(0), t] \end{aligned} \tag{3}$$

The output layer just has one neuron that corresponds to the estimated aging-induced delay degradation of a cell, i.e., $D(t) = Y$. We use three hidden layers in our FFNN and each hidden layers consists of 256 neurons. The relation between the output Y and input feature vector X can be expressed as:

$$Y = W_4 \cdot \left(f^{(3)}\left(W_3 \cdot \left(f^{(2)}\left(W_2 \cdot f^{(1)}\left(W_1 \cdot X + b_1\right) + b_2\right)\right) + b_3\right)\right) \tag{4}$$

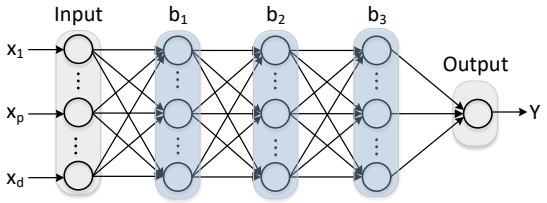


Figure 2: Architecture of our Feed-Forward Neural Network. The input corresponds to the input operating conditions and physical parameters of a cell, and the output corresponds to the estimated aging-induced delay degradation of the cell.

where W_1, W_2, W_3, and W_4 are the weights matrices of each layer, b_1, b_2, and b_3 are the bias vectors of each hidden layer, which need to be trained for the FFNN. $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ are three nonlinear functions; in this paper, we use the ReLU (Rectified Linear Units) nonlinear activation functions.

For each cell in the cell library, we will train one such FFNN model to learn the relation between the input operating conditions and physical parameters of the cell and the output delay degradation of the cell. Then for each gate in the circuit, we can use our trained FFNN models to inference its aging-induced delay degradation. To train each FFNN model using the aging-aware cell delay dataset from Section 3.1, we take the following steps:

1. *Normalization of input feature vector X.* As described in Equation 3, each feature x_i in our input feature vector X is on a different scale. For example, the temperature T ranges from $20°C$ to $120°C$, while the signal probability λ ranges from 0 to 1. Such different scaling of the features makes the FFNN training more difficult and it would make the trained model more dependent on the choice of units used in the input features. To address this issue, we normalize our input features from the training

data set using the normalization technique [33], i.e.,

$$\forall\, X \in inputSet:\ x'_i = \frac{x_i - \mu_X}{\sigma_X} \tag{5}$$

where x'_i denotes the normalized value of the i^{th} feature in input feature vector X. μ_X and σ_X respectively represent the mean and standard deviation of the input feature vector X.

2. *Initialization of weight matrices and bias vectors.* We set the weight matrices (W_1, W_2, W_3, and W_4) according to Xavier initialization [15], and initialize the bias vectors (b_1, b_2, and b_3) of the hidden layers to zero.
3. *Fine tuning using back propagation.* Finally, we fine tune the weight matrices and bias vectors, so that the predicted delay degradation Y from our FNN model approaches to the actual characterized delay degradation $\hat{Y}$ from Section 3.1. To calculate the difference between Y and $\hat{Y}$, we use Mean Squared Error (MSE) for N sample training datasets, i.e.,:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left\|Y_i - \hat{Y}_i\right\|^2 \tag{6}$$

To minimize the MSE, we use standard back propagation with the gradient descent algorithm [8].

3.3 Aging-Aware Static Timing Analysis

Timing analysis is the process of determining the circuit delay and analyzing its timing issues. There are two major approaches for timing analysis of a circuit at logic-level: *timing simulation* and *Static Timing Analysis (STA)*. Timing simulation is done by exercising the circuit using a set of input vectors and gates' parameters [11]. The comprehensiveness of this method depends on the number of input vectors and gates' parameters used for simulation. As a result, it is almost infeasible to do a complete verification of all timing constraints of modern industrial circuits which may have more than 100 million gates. In contrast, STA is carried out statistically which makes it independent of the number of input vectors and gates' parameters [11, 21]. By analyzing the entire circuit once, it provides a simpler and faster way to analyze the timing of the circuit.

A standard STA tool [11, 21] requires the following inputs from a circuit: 1) the gate-level circuit netlist, 2) the parasitic information contained in the Standard Parasitic Exchange Format file (SPEF file), and 3) the timing table of the cells. Logic level STA utilizes the timing data obtained from annotations in standard delay format, and establishes look-up tables of gate delays to quickly retrieve a constant propagation delay. In these tools, the delay of the circuit is a function of the input slew range and output load capacitance (for a specific temperature).

Incorporating aging-aware analysis into a standard STA tool makes the complexity of the timing description for the cells grow quickly, due to a versatile set of aging factors that affect the delay degradation as summarized in Section 2.1.2. Therefore, we modify the standard STA toolflow to integrate our trained FFNN models for fast aging-induced delay inference. The revised aging-aware STA flow is shown in the right part of Figure 1, with the following two major changes. First, the gate-level netlist of the circuit is fed to a logic simulator to compute the signal probability (λ) profiles according to the running workload. This provides the signal probability (λ) of each transistor inside each gate, which affects the aging-induced delay degradation. Second, the trained FFNN models for each library cell are used inside the STA tool to infer the aging-induced delay of the circuit at the projected lifetime (t). Note that all other aging factors are already available in the standard STA tool and do not need a separate extraction.

While our aging-aware STA toolflow is generic for any standard STA tools, for illustration purpose, we have used a state-of-the-art STA tool called OpenTimer [21] for the integration. The original OpenTimer is unaware of aging effects but uses a block-based approach for fast STA. As a result, our aging-aware STA flow based on OpenTimer can avoid the slow speed issue of prior critical-path based approaches summarized in Section 2.2.3 and 2.2.4, and achieve fast static timing analysis for large-scale industrial circuits (results in Section 4.5).

4 EXPERIMENTAL RESULTS

4.1 Experimental Setup

We use Google's Tensorflow [17] machine learning framework to build our proposed FFNN for aging-aware delay modeling. As described in Section 3.2, our FFNN has three hidden layers, each of which has 256 neurons. It is implemented in Python and runs on a desktop machine with an Intel Core i7 (2.5 GHz) processor and 8GB of DRAM. To get the training and testing datasets, we use the following setup.

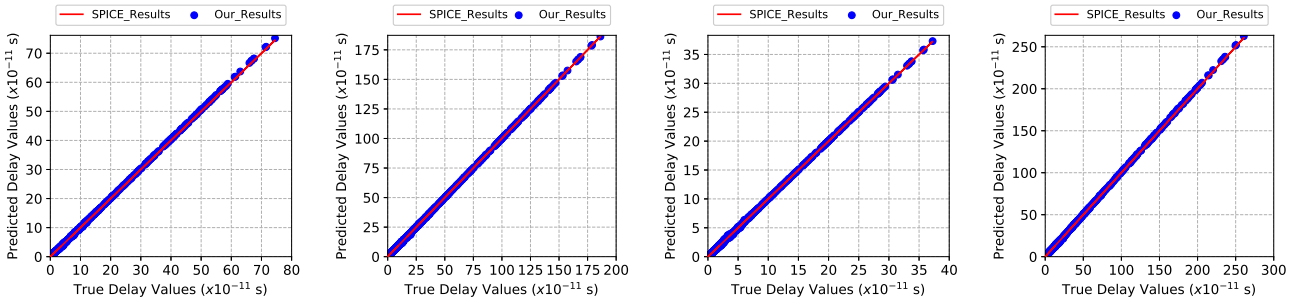
At the transistor-level, we use the 45nm predictive technology model [41] for both NMOS and PMOS transistors. The HSPICE MOSFET Model Reliability Analysis (MOSRA) [35, 36] has been used under different working temperatures (T), signal probabilities (λ), initial threshold voltage ($V_{th}(0)$) values, transistor W/L ratios, and projected lifetime (t). The ranges for the variables are: T: [20:120]°C, λ: [0:1], $V_{th}(0)$: [-0.5:-0.1] V for PMOS transistors and [0.1:0.5] V for NMOS transistors, and t: [0:10] years. We also consider 6 different W/L ratios: 1X (minimum sized), 2X, 4X, 8X, 16X, 32X (maximum sized).

At the gate-level, we use the open-source Nangate 45nm generic cell library [31] and its SPICE netlists to get realistic netlists of different combinations of cells (gates). Parasitic information is included based on the layout at the 45nm technology node. The S_{min} and S_{max} are set to 5ps and 950ps, respectively. The C_{min} and C_{max} are set to 0.25fF and 25fF, respectively. The supply voltage (V_{dd}) is set to 1.1V and the HSPICE tool is used to measure gate delays.

The dataset used for each library cell can be denoted as $dataPair = \left[X^i, \hat{Y}^i\right]$, where i denotes the sample index in the dataset, X^i denotes the input feature vector defined in Equation 3, and $\hat{Y}^i$ denotes the corresponding lifetime delay calculated by HSPICE simulation (i.e., the ground true label for our FFNN). For each library cell, we first randomly shuffle all its datasets and then split them into two disjoint sets: the training set and the validation set. In this paper, we use 1000 samples for the training dataset. For the results presented in Section 4.2 and 4.3, we use 600 samples for the validation set.

4.2 Validation for Our Cell Delay Model

We verify the accuracy of our proposed model by comparing the predicted delay of a gate with the true delay obtained from SPICE



(a) NAND (b) NOR (c) NOT (d) XNOR

Figure 3: Accuracy comparison of our predicted gate delay to the true delay of SPICE simulation

Table 1: Runtime comparison of our model compared to HSPICE MOSRA for NOT, NAND, NOR, and XNOR gates

Gate	**Our Model**	SPICE Simulation
NAND	7.14×10^{-4}s	7.05s
NOR	7.61×10^{-4}s	7.31s
NOT	7.55×10^{-4}s	5.70s
XNOR	7.74×10^{-4}s	13.97s
Average	$\mathbf{7.51 \times 10^{-4}}$**s**	8.48s

simulation. We use 600 samples for the validation. Figure 3 illustrates the scatter plots of our FFNN prediction results for NAND, NOR, NOT, and XNOR gates. The x-axis and red line show the true delay values from the SPICE results. The y-axis and blue scatter dots show the predicted delay value from our FFNN model. The closer the blue scatter dots is to the red line (i.e., the $y = x$ line), the more accurate our predicted results are. As shown in Figure 3, our FFNN models are very accurate, with predicted delay values at gate-level almost identical to the simulated delay values at transistor-level.

Moreover, as shown in Table 1, on average, our FFNN model prediction is about 10^4x faster than the transistor-level SPICE simulation. The runtime of the training phase of the proposed method for each cell is less than 3,500s.

4.3 Comparison to LUT- & SVM-based Models

To demonstrate the effectiveness of our proposed model, we also implement the LUT-based (LookUp Table) model [2] as discussed in Section 2.2.2 and an extension of the SVM-based (Support Vector Machine) model discussed in Section 2.2.4, and compare our model to them. For the SVM-based model, we first extended the model to capture the versatile aging factors discussed earlier, and then trained the SVM model using the same 1000-samples training set. Finally, we applied the trained SVM model to the same 600-samples validation set for inference. For the LUT-based model, creating a comprehensive LUT for all combinations of the aging factor values is very time consuming, since it is based on the SPICE simulation which is very slow. For a fair comparison, we use the same 1000-samples training set to build the LUT. For each input aging factor vector in the same 600-samples test set, we first search the LUT to find the exact matching entry. If it is not found in the LUT, we find the entries that have the closest aging factor combination to the input one, and then use the interpolation value of those delay values of nearby entries as the final delay value.

Table 2: Model accuracy comparison for primitive gates using RMSPE: the lower, the more accurate

Gate	**Our Model**	LUT Model [2]	SVM Model [38, 39]
NOT	0.34	3.78	1.11
NAND	0.31	4.10	1.08
NOR	0.25	3.63	1.37
AND	0.29	3.96	0.87
OR	0.26	3.81	1.07
XOR	0.18	4.95	1.12
XNOR	0.27	3.56	0.93

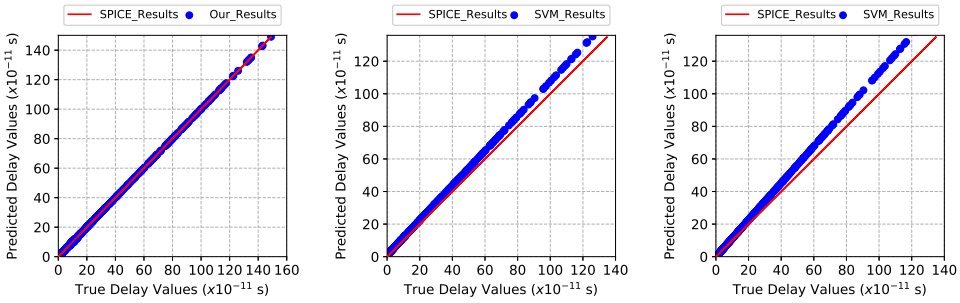
To compare the prediction accuracy of these models, we calculate the Root Mean Squared Percentage Error (RMSPE) between the predicted delay values (Y_i) and true delay values ($\hat{Y}_i$) from the SPICE simulation. RMSPE is a robust indicator of the accuracy for predicted delay values [20] and can be computed as:

$$RMSPE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\frac{Y_i - \hat{Y}_i}{Y_i}\right\|^2} \times 100 \quad (7)$$

where N denotes the number of test samples, and i denotes the i^{th} sample. The lower RMSPE is, the more accurate the model is.

Table 2 compares the obtained RMSPE results of the three models for a set of primitive gates. Compared to the LUT-based model and SVM-based model, our model achieves lower prediction errors. The LUT-based model achieves the worst accuracy since it cannot accurately predict for the input feature vectors that fall outside the pre-built LUT entries. Our FFNN based model achieves better accuracy than the SVM-based model because the FFNN learns more adaptive bias functions than the SVM. To cast more light on the prediction accuracy of these models, we also include a scatter plot for the AND gate in Figure 4. It shows that our predicted delay values are much closer to the true delay value from SPICE simulation, compared to the LUT-based model and SVM-based model.

Finally, we also compare the average speed of the three models in Table 3. Although our model is about 2.2x and 3x slower than the LUT-based model and SVM-based model, their execution times are on the same $10^{-4}s$ order.



(a) Our Model (b) LUT Model (c) SVM Model

Figure 4: Model prediction accuracy comparison for AND Gate: a) Our Model, b) LUT Model and c) SVM Model

Table 3: Model runtime comparison for primitive gates

Gate	**Our Model**	LUT Model [2]	SVM Model [38, 39]
NOT	7.55×10^{-4}s	3.12×10^{-4}s	2.62×10^{-4}s
NAND	7.14×10^{-4}s	3.85×10^{-4}s	2.67×10^{-4}s
NOR	7.61×10^{-4}s	3.53×10^{-4}s	2.37×10^{-4}s
AND	7.32×10^{-4}s	3.41×10^{-4}s	2.32×10^{-4}s
OR	7.78×10^{-4}s	3.42×10^{-4}s	2.68×10^{-4}s
XOR	7.13×10^{-4}s	3.26×10^{-4}s	2.19×10^{-4}s
XNOR	7.74×10^{-4}s	3.08×10^{-4}s	2.57×10^{-4}s
Average	7.50×10^{-4}s	3.34×10^{-4}s	2.50×10^{-4}s

4.4 Accuracy for Circuit-Level STA

4.4.1 Overall Circuit Lifetime Delay Prediction. To evaluate the effectiveness of our proposed model for predicting the lifetime delay of a circuit, we calculate its *cumulative RMSPE* that includes the accumulation and propagation of all errors for estimating the delay of each individual gate throughout the circuit. The *cumulative RMSPE* is calculated by comparing our predicted circuit lifetime delay with the true circuit lifetime delay from SPICE simulation using Equation 7. Since the SPICE simulation is very time consuming, we only demonstrate the prediction accuracy for a few small circuits from ISCAS'85 [10], ISCAS'89 [9], and ITC'99 [12] benchmark suites.

We also compare our proposed model to the critical path based approach [20] that uses a non-linear regression model for the lifetime delay prediction, which is discussed in Section 2.2.3 and 2.2.4. For the path-based model [20], to compute the circuit lifetime delay in a tractable manner, we first extract the top 20% critical paths of each circuit. Then we randomly select 1000 sample paths from these critical paths and run SPICE simulations to calculate each path's true lifetime delay. We train the the non-linear regression model used in [20] using the 1000 samples. Finally, we use the trained model to predict the lifetime delay of each path ($Delay_{Path}$) in the top 20% critical paths. We compute the lifetime delay of the circuit ($Delay_{Circuit}$) as the maximum delay of all these paths:

$$Delay_{Circuit} = \max_{1 \le i \le |Paths|} \{Delay_{Path(i)}\} \quad (8)$$

Note that our model is built on top of the block-based STA tool OpenTimer [21], so we avoid the issue of calculating for a large number of paths in the path-based model.

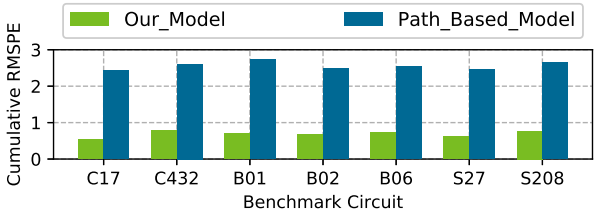
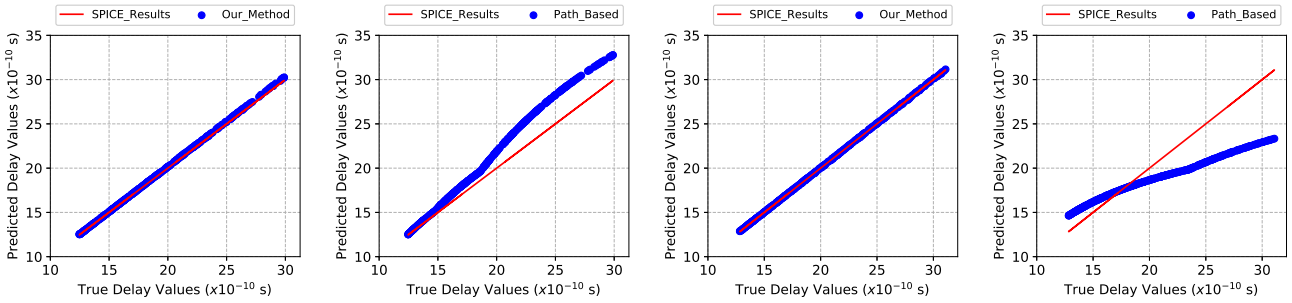


Figure 5: Accuracy comparison of our model and path-based model [20] for aging-aware static timing analysis: the lower the y-axis (cumulative RMSPE), the more accurate the model

Figure 5 compares the accuracy of our model and the path-based model [20]. The y-axis shows the average result of cumulative RMSPEs for predicting the lifetime delay of each circuit under 10 different combinations of aging factors. Compared to the path-based model, our model achieves much lower cumulative RMSPEs and is more accurate in predicting the circuit lifetime delay.

4.4.2 Path Lifetime Delay Prediction with Perturbation. To better understand the accuracy difference of our model and the path-based model [20], we also compare their accuracy when predicting the aging-induced delay for an individual path. Moreover, we also evaluate how these two models perform when one or more perturbations are made to the circuit; a good model should be stable to provide high prediction accuracy after a circuit perturbation. To perform these analysis, we randomly choose five different paths from the C880, C1908, C3540, C5315 benchmark circuits [10] and estimate each path's lifetime delay. For the circuit perturbation, we randomly changed the transistor W/L ratio for a number of gates in the path.

Table 4 compares the cumulative RMSPE of our model and the path-based model [20] for the five randomly selected paths, both before and after the perturbation. First, before the perturbation, our model achieves better accuracy than the path-based model, which accumulates to the larger accuracy gap at the circuit level as presented in Figure 5 of Section 4.4.1. Second, after the perturbation, our model's accuracy is stably high, while the path-based model's accuracy significantly decreases. The main justification is that the coarse-grained prediction of a path (i.e., path-based model) is harder than the fine-grained prediction of a gate (i.e., block-based model in our tool). As a result, after a perturbation, the path-based model may need a model retraining to improve its accuracy, while our model does not need model retraining.



(a) Our Model before Perturbation **(b) Path-based Model before Perturbation** **(c) Our Model after Perturbation** **(d) Path-based Model after Perturbation**

Figure 6: Comparison of accumulated error for predicting the aging-induced delay for path P1 with perturbation to the path

Table 4: Model accuracy comparison for predicting the aging-induced delay for each path with perturbation to the path. Prediction error is in terms of cumulative RMSPE: the lower, the better

Path	# Gates	Before Perturbation		After Perturbation	
		Our Model	Path-based Model [20]	**Our Model**	Path-based Model [20]
P1	38	0.86	1.57	0.78	5.26
P2	37	0.64	1.49	0.74	4.79
P3	34	0.71	1.52	0.86	4.33
P4	35	0.67	1.38	0.61	4.86
P5	40	0.79	1.47	0.83	4.51

Table 5: Manageable runtime of our aging-aware STA compared to standard aging-unaware STA for large-scale circuits

Circuit	# Gates	**Our Aging Aware STA**	OpenTimer: Aging Unaware STA [21]
des_perf [21]	138.9K	28.01s	22.96s
vga_lcd [21]	139.5K	28.07s	23.05s
b19 [12]	255.3K	53.71s	41.87s

To cast more light on the path delay prediction using our model and the path-based model before and after the perturbation, we also include a scatter plot for path P1 in Figure 6. As shown in Figure 6a and Figure 6c, our model can accurately predict the path delay both before and after the perturbation, which is close to the SPICE simulation results. On the other hand, the path-based model already introduces quite some errors before the perturbation, as shown in Figure 6b. After the perturbation, as shown in Figure 6d, the predicted delays of the path-based model are significantly different to the SPICE simulation results.

4.5 STA Runtime for Large-Scale Circuits

Finally, to demonstrate the runtime efficiency of our aging-aware static timing analysis (STA) flow that integrates our FFNN models, we also calculate the circuit delay of three large-scale circuits from ITC'99 [12] and OpenTimer [21] benchmarks. For the standard aging-unaware STA tool, we use a state-of-the-art STA tool called OpenTimer [21]. As presented in Section 3.3, our aging-aware STA is built on top of OpenTimer with the integration of our FFNN models to predict the aging-induced gate delays. Table 5 compares the runtime of our tool to the original OpenTimer: our aging-aware delay calculation only adds around 5 to 12 seconds extra runtime for a circuit with 138.9K to 255.3K gates, which is manageable for large-scale circuit analysis.

5 CONCLUSION

Bringing accurate and fast aging-aware timing analysis to existing EDA toolflows is essential in obtaining reliable large-scale circuit designs. In this paper, we present Aadam, a fast, accurate, and versatile aging-aware delay model for generic cell libraries, and integrate it with the widely used open-source static timing analysis (STA) tool called OpenTimer. Aadam characterizes the delay degradation of each library cell under a wide range of aging factors with the transistor-level SPICE simulation. Based on the characterized data, Aadam trains an FFNN to learn the relation between the input aging factors and output delay degradation for each library cell. As a result, the trained FFNNs inside Aadam can quickly and accurately predict the gate and circuit lifetime delay degradations under a versatile combination of aging factors. Experimental results demonstrate that Aadam achieves almost identical accuracy to the transistor-level simulation in predicting the aging-induced delay of a circuit, which is more accurate than prior lookup table based model, support vector machine based model, and critical path based model. Moreover, it enables aging-aware STA for large-scale circuits with manageable runtime overhead, i.e., 5 to 12 seconds runtime overhead to analyze a circuit with 138.9K to 255.3K gates. We plan to open source our entire Aadam toolflow in the near future to trigger more research in applying machine learning techniques to aging-aware circuit analysis.

ACKNOWLEDGEMENTS

We acknowledge the support from Government of Canada Technology Demonstration Program and MDA Systems Ltd; Natural Sciences and Engineering Research Council of Canada (NSERC Discovery Grant RGPIN-2019-04613 and DGECR-2019-00120); Canada Foundation for Innovation John R. Evans Leaders Fund; Simon Fraser University New Faculty Start-up Grant; Xilinx, Huawei and Nvidia.

REFERENCES

[1] M Alam, K Kang, BC Paul, and K Roy. 2007. Reliability-and process-variation aware design of vlsi circuits. In *2007 14th International Symposium on the Physical and Failure Analysis of Integrated Circuits*. IEEE, 17–25.

[2] Hussam Amrouch, Behnam Khaleghi, Andreas Gerstlauer, and Jörg Henkel. 2016. Reliability-aware design to suppress aging. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[3] Hussam Amrouch, Javier Martin-Martinez, Victor M van Santen, Miquel Moras, Rosana Rodriguez, Montserrat Nafria, and Jörg Henkel. 2015. Connecting the physical and application level towards grasping aging effects. In *2015 IEEE International Reliability Physics Symposium*. IEEE, 3D–1.

[4] Hussam Amrouch, Victor M van Santen, Thomas Ebi, Volker Wenzel, and Jörg Henkel. 2014. Towards interdependencies of aging mechanisms. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 478–485.

[5] W. T. Anderson. 2001. Semiconductor device reliability in extreme high temperature space environments. In *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, Vol. 5. 2457–2462 vol.5.

[6] Senthil Arasu, Mehrdad Nourani, John M Carulli, and Vijay K Reddy. 2015. Controlling aging in timing-critical paths. *IEEE Design & Test* 33, 4 (2015), 82–91.

[7] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. 2003. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th annual Design Automation Conference*. 338–342.

[8] Léon Bottou. 1998. Online learning and stochastic approximations. *On-line learning in neural networks* 17, 9 (1998), 142.

[9] F. Brglez, D. Bryan, and K. Kozminski. 1989. Combinational profiles of sequential benchmark circuits. In *Circuits and Systems, 1989., IEEE International Symposium on*. 1929–1934 vol.3.

[10] F. Brglez and H. Fujiwara. 1985. A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran. In *Proceedings of IEEE Int'l Symposium Circuits and Systems (ISCAS 85)*. IEEE Press, Piscataway, N.J., 677–692.

[11] Rakesh Chadha and J Bhasker. 2009. *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer.

[12] F. Corno, M. S. Reorda, and G. Squillero. 2000. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design Test of Computers* 17, 3 (July 2000), 44–53.

[13] Balbir S Dhillon. 2000. *Medical device reliability and associated areas*. CRC Press.

[14] Mojtaba Ebrahimi, Fabian Oboril, Saman Kiamehr, and Mehdi B Tahoori. 2013. Aging-aware logic synthesis. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 61–68.

[15] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[16] Dennis Gnad, Muhammad Shafique, Florian Kriebel, Semeen Rehman, Duo Sun, and Jörg Henkel. 2015. Hayat: Harnessing dark silicon and variability for aging deceleration and balancing. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[17] Google. 2020. Tensorflow: An end-to-end open source machine learning platform. (2020). https://www.tensorflow.org/

[18] Marco Gori. 2018. Chapter 5 - Deep Architectures. In *Machine Learning*, Marco Gori (Ed.). Morgan Kaufmann, 236 – 338. http://www.sciencedirect.com/science/article/pii/B9780081006597000051

[19] Jörg Henkel, Lars Bauer, Nikil Dutt, Puneet Gupta, Sani Nassif, Muhammad Shafique, Mehdi Tahoori, and Norbert Wehn. 2013. Reliable on-chip systems in the nano-era: Lessons learnt and future trends. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 1–10.

[20] Ke Huang, Xinqiao Zhang, and Naghmeh Karimi. 2019. Real-Time Prediction for IC Aging Based on Machine Learning. *IEEE Transactions on Instrumentation and Measurement* 68, 12 (2019), 4756–4764.

[21] Tsung-Wei Huang and Martin DF Wong. 2015. OpenTimer: A high-performance timing analysis tool. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 895–902.

[22] Andrew B Kahng. 2018. New directions for learning-based IC design tools and methodologies. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 405–410.

[23] Andrew B Kahng, Uday Mallappa, and Lawrence Saul. 2018. Using Machine Learning to Predict Path-Based Slack from Graph-Based Timing Analysis. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 603–612.

[24] Mehdi Kamal, Qing Xie, Massoud Pedram, Ali Afzali-Kusha, and Saeed Safari. 2012. An efficient reliability simulation flow for evaluating the hot carrier injection effect in CMOS VLSI circuits. In *2012 IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 352–357.

[25] Naghmeh Karimi and Ke Huang. 2016. Prognosis of NBTI aging using a machine learning scheme. In *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 7–10.

[26] Naghmeh Karimi, Arun Karthik Kanuparthi, Xueyang Wang, Ozgur Sinanoglu, and Ramesh Karri. 2015. Magic: Malicious aging in circuits/cores. *ACM Transactions on Architecture and Code Optimization (TACO)* 12, 1 (2015), 1–25.

[27] John Keane and Chris H Kim. 2011. Transistor aging. *IEEE Spectrum* 48, 5 (2011), 28–33.

[28] Saman Kiamehr, Farshad Firouzi, Mojtaba Ebrahimi, and Mehdi B Tahoori. 2014. Aging-aware standard cell library design. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–4.

[29] Saman Kiamehr, Farshad Firouzi, and Mehdi B Tahoori. 2013. Aging-aware timing analysis considering combined effects of NBTI and PBTI. In *International Symposium on Quality Electronic Design (ISQED)*. IEEE, 53–59.

[30] Dominik Lorenz, Georg Georgakos, and Ulf Schlichtmann. 2009. Aging analysis of circuit timing considering NBTI and HCI. In *2009 15th IEEE International On-Line Testing Symposium*. IEEE, 3–8.

[31] Mayler Martins, Jody Maick Matos, Renato P Ribas, André Reis, Guilherme Schlinker, Lucio Rech, and Jens Michelsen. 2015. Open cell library in 15nm FreePDK technology. In *Proceedings of the 2015 Symposium on International Symposium on Physical Design*. 171–178.

[32] Spencer Millican, Yang Sun, Soham Roy, and Vishwani Agrawal. 2019. Applying Neural Networks to Delay Fault Testing: Test Point Insertion and Random Circuit Training. In *2019 IEEE 28th Asian Test Symposium (ATS)*. IEEE, 13–135.

[33] Pierre Sermanet, Soumith Chintala, and Yann LeCun. 2012. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 3288–3291.

[34] Robert H Tu, Elyse Rosenbaum, Wilson Y Chan, Chester C Li, Eric Minami, Khandker Quader, Ping K Ko, and Chenming Hu. 1993. Berkeley reliability tools-BERT. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 12, 10 (1993), 1524–1534.

[35] Bogdan Tudor, Joddy Wang, Weidong Liu, and Hany Elhak. 2011. MOS device aging analysis with HSPICE and CustomSim. *Synopsys, White Paper* (2011). https://www.synopsys.com/content/dam/synopsys/verification/white-papers/mosra-wp.pdf

[36] Bogdan Tudor, Joddy Wang, Charly Sun, Zhaoping Chen, Zhijia Liao, Robin Tan, Weidong Liu, and Frank Lee. 2010. MOSRA: An efficient and versatile MOS aging modeling and reliability analysis solution for 45nm and below. In *2010 10th IEEE International Conference on Solid-State and Integrated Circuit Technology*. IEEE, 1645–1647.

[37] Jyothi Bhaskarr Velamala, Venkatesa Ravi, and Yu Cao. 2011. Failure diagnosis of asymmetric aging under NBTI. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 428–433.

[38] Arunkumar Vijayan, Krishnendu Chakrabarty, and Mehdi B Tahoori. 2019. Machine Learning-Based Aging Analysis. In *Machine Learning in VLSI Computer-Aided Design*. Springer, 265–289.

[39] Arunkumar Vijayan, Abhishek Koneru, Saman Kiamehr, Krishnendu Chakrabarty, and Mehdi B Tahoori. 2016. Fine-grained aging-induced delay prediction based on the monitoring of run-time stress. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 5 (2016), 1064–1075.

[40] Wenping Wang, Shengqi Yang, Sarvesh Bhardwaj, Rakesh Vattikonda, Sarma Vrudhula, Frank Liu, and Yu Cao. 2007. The impact of NBTI on the performance of combinational and sequential circuits. In *Proceedings of the 44th annual Design Automation Conference*. 364–369.

[41] Wei Zhao and Yu Cao. 2006. New generation of predictive technology model for sub-45 nm early design exploration. *IEEE Transactions on Electron Devices* 53, 11 (2006), 2816–2823.