SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Machine Learning-Based Static Timing Analysis for Advanced Semiconductor Design: Approaches, Challenges, and Solutions

Sagar Mallik

Independent Researcher, USA

Abstract
This article discusses the growing performance and energy efficiency challenges with STA needed for sub-10nm technology nodes and the associated resource limitations of conventional EDA tools in achieving timing closure. With machine learning techniques, important cell-arc and wire delay estimation speedups are achieved through the training of predictive machine learning models over design data with representative characteristics. This enables wire delay estimation with a high degree of accuracy and within acceptable limits. The two main challenges in wire delay estimation methods are global routing versus detailed routing miscorrelation and parasitic parameter estimation. Process variation modeling of devices must include SPICE-trained cell delay models in addition to variations in parameters such as crosstalk, temperature, and voltage variations, and distance-based variation, and may employ a combined neural network. In addition, Electromigration (EM) induced self-heating, and IR drop hotspots can be predicted by ML models trained on proper design features. Depending on the requirements, such as speed, accuracy, and availability of hardware resources, the regression may be performed using linear regression, random forest ensemble regression, or deep neural networks. Employing these regression models has resulted in run-times on ECO phases exceeding 1000 times faster march to tape-out and signoff.

Keywords: Static Timing Analysis, Machine Learning, Delay Estimation, Wire Parasitic Prediction, Process Variation Modeling, Neural Networks, Timing Closure, Advanced Semiconductor Nodes, Engineering Change Order (ECO), Graph Based Analysis (GBA), Path Based Analysis (PBA)

1. Introduction

Modern sub-10 nm technology node semiconductor designs are constrained by compute costs and runtime limits for static timing analysis (STA). Existing electronic design automation (EDA) tools struggle on large system on chip (SoC) designs. These challenges are heightened in high-performance compute (HPC) designs that approach the reticle limit and have billions of signal transitions. Advanced EDA tools must be able to accommodate advanced delay computation techniques such as Advanced Waveform Propagation (AWE), crosstalk, and Multi-Input Switching (MIS) methods, as well as advanced input models, including Constant Current Source (CCS) modeling, LVF, aging and distance-based derating factors.

As a result of the increasing load from timing closure flows, full-flat timing analysis of reticle-size designs at medium utilization is no longer practical without additional design work, for example, partitioning schemes like hyperscale and hypergrid, resulting in wide blind spots in the analysis. Hence, the trade-off between using simplified lookup tables that over-margin designs and degrade PPA versus using accurate modeling approaches that overload the timer engines remains one of the continuing challenges for modern solutions [1].

In this context, ML methods offer an alternative approach: predictive models trained on representative designs (e.g., AI/ML matrix multiplier units, vector processing units, and CPU cores) can produce delay estimates that are achieved 1000x faster (or more) for only ~10% accuracy loss. This is especially useful in the Engineering Change Order (ECO) iteration phase, when there are many changes that need to be processed in rapid succession, whereas full signoff runs can be separated when it is actually time for tapeout [1]. This article reviews recent developments in STA based on ML, a technique that seeks to solve fundamental problems of wire delay estimation, statistical path analysis, crosstalk modeling, aging, and temperature-voltage variation modeling.

2. Contemporary STA Computational Challenges and Machine Learning Solutions

The challenges in STA at advanced nodes can be increased due to several factors. For example, the increased sensitivity to parasitic extraction at advanced nodes requires the use of advanced extraction models [2]. Secondly, the number of gates and nets in today's designs can be up to a billion, making data sets difficult

to analyze. Third, the days-long turnaround time when running ECO cycles adds months to the timing of the signoff [1].

Customary STA uses characterization-based methods, where the gate delays are characterized by simulation or measurement at all corners of the process voltage and temperature (PVT). These are expensive at the advanced technology nodes, as thousands of Monte Carlo runs are used to account for the process variation through Library Variation Format (LVF) representations. This trade-off is due to the conservative lookup tables used for performance optimization versus accurate models that are computationally expensive.

These issues are addressed by ML techniques that learn the mapping from input features (input transition times, output load capacitances, depths of stages in gates (GBA metrics), bounding boxes of timing paths (GBA metrics), and attributes of segments in the metal layers) to output labels (delays of cell arcs and wires). ML models can generalize to unseen topologies or operating conditions without requiring any explicit characterization. This approach was shown to reduce run times by more than three orders of magnitude, with a tolerable impact on accuracy needed to achieve ECO-phase timing closure.

Table 1. Input Features and Output Labels for ML-Based Delay Prediction

Input Feature Category	Specific Parameters	Physical Significance	Prediction Target
Temporal Characteristics	Input transition times	Signal edge rate influence	Cell-arc delay
Load Conditions	Output load capacitance	Downstream load impact	Cell-arc delay
Logic Depth Metrics	Stage count (GBA) through cell pin	Random Process Variation	Cell-arc delay
Geometric Constraints	Timing path bounding box dimensions (GBA)	Distance variation over a Timing Path	Cell-arc and Wire delay
Interconnect Attributes	Metal layer segment properties	Layer-dependent parasitic variation	Wire delay
Design Hotspot Data	Routing track overflow	ECOed and new Wire Topology prediction	Wire delay

3. Wire Delay Estimation: The Central Challenge of ML-Based STA

Perhaps the most challenging problem in ML-based STA is wire delay prediction, where net topology and parasitic parameters in the design change over ECO iterations [1]. The need for the design to be taken through the PNR flow for physical implementation, for multiple iterations of ECOs, adds significant delay to the schedule. Errors in wire delay estimation, especially for the eco-induced new nets, can propagate through timing paths, accumulating a chain reaction of errors in the model.

First, a global route (GR) is fast and quick and can be achieved easily with ML precision. But the estimated delays from the global route can be very different from the detailed route (DR) delay. For example, the Steiner tree-based lookup tables, such as FLUTE, have very poor accuracy in the sub-10 nm regime, since the interconnect performance becomes extremely sensitive to variations in the geometric parameters [2].

Second, the GR-versus-DR miscorrelation problem cannot be fixed by tuning parameters alone [1]. While global routing engines can compute very efficient paths through simple greedy algorithms, detailed routers must deal with resource conflicts, which leads to vastly different topologies than those predicted by the global router in many instances. This divergence may also be more pronounced in constrained regions (routing congestion hotspots) and regions near design boundary conditions [1].

To address these issues, the current approach is to move from GR-based RC delay predictors to DR-informed predictors. It has been shown that by directly training models on DR instead of GR topology information, miscorrelations are reduced [7]. For ECOed and new wires, it is important to read the design congestion hotspot (routing track overflow) as a feature to predict the actual DR path. ML prediction is used to predict the DR net topology and layer assignment information correctly with this information [Fig. 1].

Table 2. Wire Delay Estimation Challenges: Global Route versus Detailed Route [1][7]

Wire Delay Method	Algorithm Characteristics	Topology Assumptions	Accuracy Limitation	Key Miscorrelation Driver
Global Route (GR) based	Simplified and fast	Ideal interconnect paths	High inaccuracy at sub-10 nm	Neglects congestion-driven detours
Detailed Route (DR) based	Physical Shape-Based Parasitic Extraction	Actual geometric implementation	Reference ground truth	Layer assignment decisions
FLUTE Steiner Tree	Lookup table-based prediction	Mathematical optimization paths	Insufficient for advanced nodes	Assumes isotropic interconnect
GR-DR Miscorrelation	Systematic divergence source	Congestion-dependent topology shifts	Eliminable through parameter tuning alone	Design boundary condition sensitivity
Congestion Driven Routing Prediction	Wire topology and physical layer assignment model	Accurate geometric estimation	Miscorrelation < 10%	Physical Macros and EDA tool behavior

4. Statistical Path Delay and Process Variation Representation

Further improvement can be achieved by basing the reference cell delays not on pre-characterized LVF libraries but by training on simulations using SPICE models instead, which avoids pessimism due to inaccuracies in the library and avoids the cost of corner-specific lookup table-based STA [6]. Additionally, this approach couples models for self-heat and IR drop prediction trained on representative designs to predict cell delay variation at different physical locations of the design [Fig. 1].

Cell delay models can be trained directly from SPICE-level simulations. This approach has the advantage of being more representative of the underlying physical mechanisms driving delay variation, not requiring characterizations across a wide variety of corners, and avoiding the need for expensive Monte Carlo simulation over many corners. Training delay models directly from SPICE-level simulations does not demand the same library characterization infrastructure . When combined with path-depth and bounding distance-aware variation scaling, it achieves higher accuracy in statistical timing analysis.

Table 3. Process Variation Representation Methodologies

Approach	Training Data Source	Physical Mechanism Capture	Corner Requirements	Accuracy	Runtime
LVF-Based (Characterized Library)	Pre-characterized cells across corners	Indirect through library metrics	Extensive corner characterization	Accurate for stated process; pessimistic for missing corners	Slow
Voltage, Temperature, and Aging	LUT-based derating	Indirect through library metrics	Extensive corner characterization	Pessimistic	Fast
SPICE-Trained Cell Delays	Direct circuit-level simulation	Direct from the model	Drastically reduced through direct training	Accurate; includes Process, Voltage, Temperature, and Aging	Fast

5. Crosstalk, Aging, Temperature-Voltage Variation, and Distance-Based Derating

Other second-order effects also impact timing accuracy on advanced nodes. One example is crosstalk delay estimation due to capacitive coupling between neighboring nets, which can lead to dynamic delays. The ML problem is of a high degree [5] and difficult since the crosstalk effects depend sensitively on the switching patterns and net spacing and layer assignment, but the pattern and spacing can vary greatly in a design [5].

Another source of die variation is temperature and voltage variation. This kind of variation is often not captured in the customary design flow for static timing analysis (STA), and the Library Variation Format for process variation does not capture the local voltage and temperature variation due to EM and IR drop. HPC designs are characterized by running at multiple power states and multiple voltages, which create

large local variations in voltage gradients. Likewise, the heat generated in regions with high current density creates large temperature gradients on the die face. Both are systematic effects and are explicitly modeled. Distance-based variation is a systematic derating factor provided by the foundry and defines how the distance between timing paths affects the variation of the environment that the timing paths see, or how geometrically distant paths are subjected to different variations. LVF does not account for this phenomenon because it assumes variation is isotropic in the design. Distance-based derating takes direct account of path location and orientation relative to power delivery infrastructure and thermal sources.

ML models trained over these SPICE simulations can learn to capture these interdependencies naturally. By varying the supply voltage, temperature, and the coupling in the training dataset, neural network models can implicitly learn the interaction of these parameters while predicting the delay [8]. To avoid expensive Monte Carlo (MC) analysis over all voltage states and all temperature conditions at STA runtime, jitter and voltage-based delay variation can be captured at STA runtime based on path-dependent characteristics of specific paths.

Table 4. Environmental Effects in Advanced Node Timing

Effect Category	Physical Origin	Timing Impact Type	Design Dependency	Conventional Modeling Limitation
Crosstalk Delay	Capacitive coupling between adjacent nets	Dynamic switching delay shifts	Switching patterns, net spacing, layer assignments	High degree of inaccuracy in prediction
Electro Migration (EM) and Self-Heat	Current-induced voltage reduction and temperature variation	Local voltage and temperature deviation from nominal	Power delivery network proximity	Not captured in the library; ML model predicts hotspot
IR Drop	Resistive voltage loss in the power grid	Supply voltage variation across the die	Current distribution and density	Not captured in the library; ML model predicts hotspot
Temperature Gradients	Heat dissipation from logic switching	Local temperature variation	Thermal source proximity and orientation	LUT-based margining
Distance-Based Derating	Foundry-provided systematic variation	Path-metric-dependent effects	Path spreads from a common point	LUT-based margining

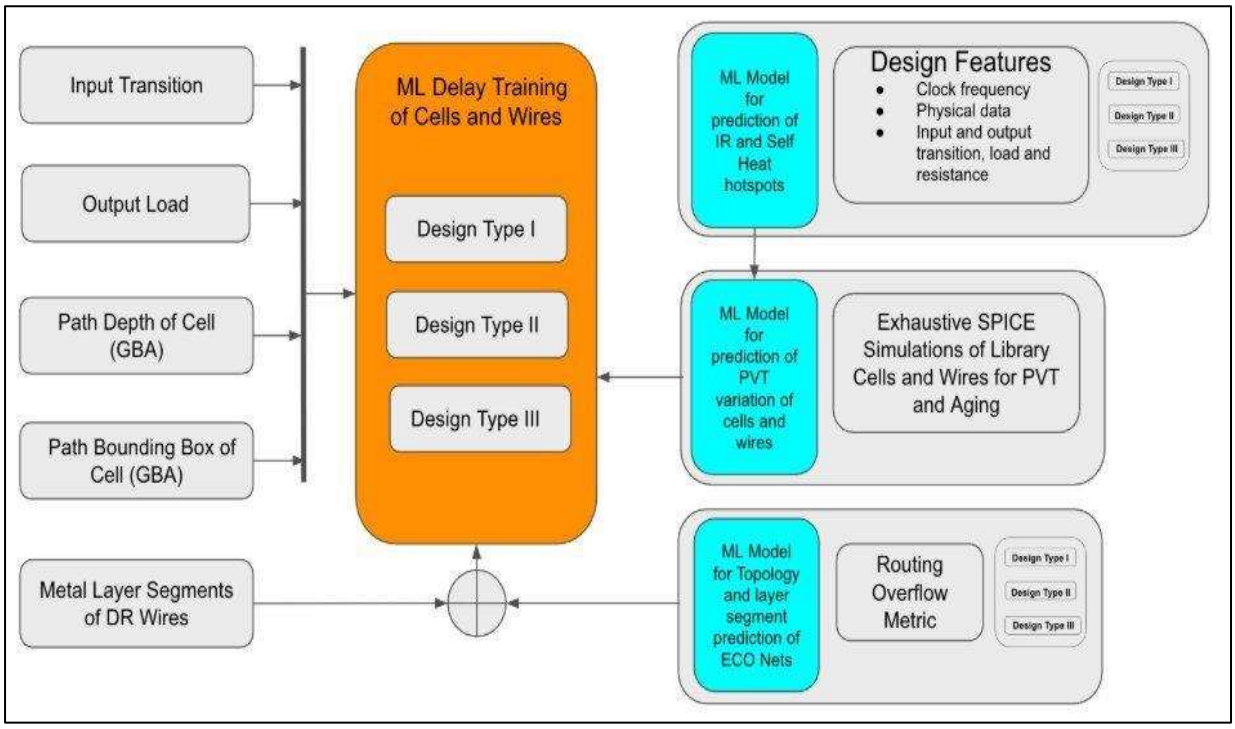

Figure 1. ML Training for Delay — Generating Standard Delay Format (SDF). Blue boxes represent Inference Models; orange boxes represent the Training Model.

6. Machine Learning Regression Approaches for Delay Estimation

The delay estimation problem can be expressed as a multivariate regression problem where the objective is to develop a function mapping a set of features (e.g., transition times, loads at cells' outputs, depths at stages, bounding boxes' dimensions, and properties of the metal segments) to the cell-arc delay and the wire delay. Different types of regression models can be applied with different trade-offs.

6.1. Linear Regression

The simplest model is linear regression, which assumes a linear relationship between the delay and the features. It incurs a low inference time and training cost but cannot learn the nonlinear relationship in the mechanisms for physical delay [3]. Linear models can be useful as a baseline or in ensemble methods.

6.2. Random Forest Regression

Random Forest regression is an ensemble technique that builds a number of decision trees and is capable of better nonlinear regression and fast inference times [3]. Random forests are also often better at automatically capturing interaction effects without much hyperparameter tuning compared to deep learning models. However, they may fail to model smoothly varying relationships and will generally need larger datasets than linear regression to be as accurate [3].

6.3. Deep Neural Networks

Fully connected DNNs with multiple hidden layers and advanced non-linear activation functions (such as ELU, the Exponential Linear Unit) are the most powerful architecture for discovering complex non-linear relations between the inputs and the delayed outputs [3]. Regularization techniques (dropout) allow a good generalization despite the large number of parameters [3]. Advanced optimization algorithms like AdamOptimizer adaptively change the learning rate during training, which helps find good solutions [3].

6.4. Ensemble Approaches

The choice of approach depends on a number of aspects of the problem being modeled, for example, the amount and quality of available training data, the required prediction accuracy, and interpretability. Ensemble models combine predictions from multiple base models, such as linear, random forest, and neural network models, to improve prediction reliability [3]. By training on SPICE-derived labels instead of library ones, all models are trained on the best available reference, preventing systematic bias from propagating.

7. Conclusion

This represents a model shift from conventional delay estimation methods to ML-based approaches, such that a faster timing closure can be achieved for HPC designs at advanced technology nodes. When these methods can achieve a more than three orders of magnitude speedup in timing closure with acceptable accuracy margins, ML-based STA is a requirement and not just an optimization for designs near and beyond the reticle size.

Key improvements include: more accurate wire delay predictions that account for detailed routing and topology after engineering change orders; process variation accounting with SPICE-trained models without pessimisms used in library characterizations; combined NN frameworks that account for crosstalk, temperature-voltage gradients, electromagnetic and IR drop, and distance-dependent derating factors; and the optimal choice among linear regression, random forest, or deep neural networks according to design constraints.

By using ensemble timing methods that combine predictions from multiple regressors, one can reduce regressor bias and obtain a higher-quality prediction. The combination of quick full-flat timing analysis on reticle-sized designs and uncompromised signoff verification runs for tape-out eliminates a major timing closure productivity bottleneck for modern semiconductor chips. Machine learning algorithms for timing analysis are becoming essential for semiconductor designs approaching exascale in complexity in order to deliver those designs on an economical timescale.

References

[1] V. A. Chhabria et al., "A Machine Learning Approach to Improving Timing Consistency between Global Route and Detailed Route," ACM Digital Library, 2023. https://dl.acm.org/doi/full/10.1145/3626959

[2] C. J. Alpert et al., "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," IEEE Trans. Comput.-Aided Design, vol. 27, no. 1, pp. 70–83, 2008.

[3] J. Xu, L. Jin, W. Fu, and L. Shi, "A Deep-learning-based Statistical Timing Prediction Method for Sub-16nm Technologies," in Proc. DATE Conf., 2024. https://past.date-conference.com/proceedings-archive/2024/DATA/135_pdf_upload.pdf

[4] P. Kannan et al., "Interconnect Estimation for FPGAs," IEEE Trans. Comput.-Aided Design, vol. 25, no. 8, pp. 1613–1625, 2006.

[5] A. B. Kahng, M. Luo, and S. Nath, "SI for Free: Machine Learning of Interconnect Coupling Delay and Transition Effects," IEEE Xplore, 2015. https://ieeexplore.ieee.org/document/7171706

[6] F. Li et al., "Prediction-Enhanced Monte Carlo: A Machine Learning View on Control Variate," arXiv:2412.11257, 2024.
[7] H.-H. Cheng, I. H.-R. Jiang, and O. Ou, "Fast and Accurate Wire Timing Estimation on Tree and Non-Tree Net Structures," IEEE Xplore, 2020. https://ieeexplore.ieee.org/document/9218712
[8] T. Martin et al., "A Machine Learning Approach to Predict Timing Delays During FPGA Placement," in Proc. IEEE FPT, 2021. https://ieeexplore.ieee.org/document/9460398
[9] K. Mistry et al., "A 45nm Logic Technology with High-k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers," in Proc. IEEE IEDM, 2007.
[10] A. Daundkar, "Error Rate Estimation in CMOS Circuits: A TCAD and SPICE-Based Approach," IJECET, 2024.
[11] B. Li et al., "Reliability Challenges for Copper Interconnects," Microelectron. Reliab., vol. 44, pp. 3–11, 2004.
[12] A. B. Kahng et al., "Machine Learning Applications in Physical Design: Recent Results and Directions," in Proc. ACM ISPD, 2018.
[13] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron," in Proc. IEEE/ACM ICCAD, 1998.
[14] P. Friedberg et al., "Modeling Within-Die Spatial Correlation Effects for Process-Design Co-optimization," in Proc. IEEE ISQED, 2005.
[15] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. New York: Springer, 2009.
[16] I. H. Sarker, "Deep Learning: A Comprehensive Overview," SN Comput. Sci., vol. 2, no. 6, p. 420, 2021.
[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, pp. 436–444, 2015.