Application of Deep Learning in Back-End Simulation: Challenges and Opportunities

Yufei Chen[1], Haojie Pei[2], Xiao Dong[1], Zhou Jin[2], Cheng Zhuo[1*]
[1]College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou, China
[2]College of Information Science and Engineering, China University of Petroleum, Beijing, China
[*]Email: czhuo@zju.edu.cn

***Abstract*—Relentless semiconductor scaling and ever increasing device integration have resulted in the exponentially growing size of the back-end design, which makes back-end simulation very time- and resource-consuming. With the success in the computer vision community, deep learning seems a promising alternative to assist the back-end simulation. However, unlike computer vision tasks, most back-end simulation problems are mathematically and physically well-defined, e.g., power delivery network sign off and post-layout circuit simulation. It then brings broad interests in the community where and how to deploy deep learning in the back-end simulation flows. This paper discusses a few challenges that the deployment of deep learning models in back-end simulation have to confront and the corresponding opportunities for future research.**

I. Introduction

Relentless semiconductor scaling and ever increasing SoC integration have imposed various challenges to the back-end design from problem size to complexity [1]. In order to understand how the design behaves when implemented in physical layout, designers have to perform a back-end simulation from the extracted view. With the smaller geometry and increased number of transistors, back-end simulation has observed fast growing circuit size and the exponential increase in interconnect parasitics, which makes back-end simulation extremely time- and resource-consuming [2].

As an essential step to validate system performance or ensure chip sign off, back-end simulations can be roughly categorized into linear and nonlinear simulations. For example, power delivery network (PDN) sign off is to solve a huge sparse linear system in the end, i.e., $Ax = b$, where A is the system matrix with a significantly large portion of zeros, b is the excitation vector, and x is the node voltage vector to be solved [3]. In modern VLSI designs, such a linear system can easily have billions of nodes with various sparsity patterns [4], incurring significant computational and memory overheads. Moreover, due to the varying test vectors, i.e., excitations, the power delivery linear system needs to be repeatedly simulated tens to even hundreds of times to ensure power integrity, which makes power delivery sign off very time consuming [4], [5]. Post-layout circuit simulation is a common nonlinear simulation example for mixed-signal and analog circuit verification [6]. Thanks to the growing parasitics impact, it has become a huge risk to simply simulate the schematic design without the layout parasitics. However, as the SPICE simulation needs to go down to the transistor level to replicate the exact model of circuit behavior, the nonlinear circuit simulation, along with large parasitics network, may take days to complete and have become a bottleneck to the verification [7]. To address the inefficiency issues in back-end simulators, researchers have proposed many physical-law or heuristics based strategies from different modeling, ordering, factorization, to solver, so as to handle the huge size of back-end problems [7]–[9]. However, as the device non-linearity continues grows and the number of parasitics exponentially increases, the gains from many prior heuristics quickly fall behind the demands of the back-end simulations [10].

On the other hand, the success of artificial intelligence (AI), especially deep learning, in the computer vision tasks, has brought broad interests in applying AI to back-end problems. A recent example is Google's Placer [11] using deep reinforcement learning followed by heated discussions in both academia and industry. It should be noted that, unlike many computer vision problems, most back-end simulation problems already have a theoretically sound mathematical and physical law-based formulations [6]. This then brings a natural question on the scope and effectiveness that deep learning techniques may bring to the back-end simulations. Unlike many deep learning based works on back-end physical design, e.g., placement and routing, which are found to a promising alternative, there are actually limited deep learning based back-end simulation that can achieve both efficiency and generality [12]–[17]. For example, a few prior works treated the power delivery noise map as an image and used the convolutional neural network (CNN) to estimate the IR drop [5], [12], [18]. While the execution of CNN is faster than the linear system solver, the training cost and generality are concerning and usually limited to a particular scenario. Thus, although deep learning is effective in addressing the large scale and non-linearity in back-end simulations, the existing challenges from training, model to generality still prevent its wide deployment in the back-end simulation:

- **Training**: The deep learning model can be dedicated to a particular design and scenario. The underlying training cost from data collection to setup is often non-negligible. Designers have to evaluate the implicit cost before the deployment.
- **Model**: The back-end circuit or layout can be huge and redundant. Domain knowledge is highly desired to select

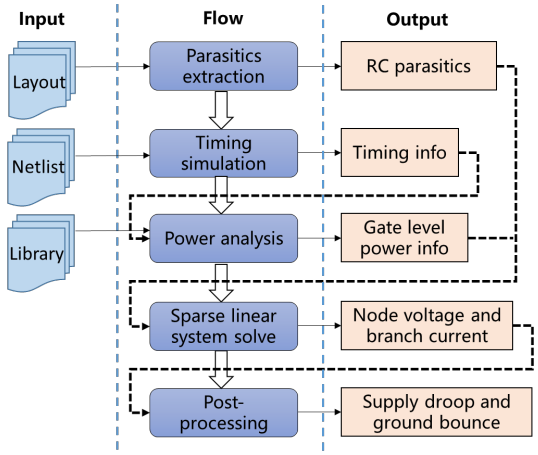

Fig. 1. Traditional PDN sign off flow.

the proper learning architecture as a trade off between efficiency and accuracy.

- **Generality**: Since the training and model design may consume non-trivial efforts, the model generality is preferred but challenging in order to support different designs, scenarios and even technologies.

In the following, we will first give a background review, and then go into details with examples for the above challenges. After that, we will conclude the paper and discuss the potential research opportunities.

II. Background

A. PDN Sign Off

Traditional PDN sign off involves multiple steps, from parasitics extraction, timing analysis, power estimation, to sparse linear system solve, as shown in Fig. 1 [4], [19]. The core problem of PDN sign off is to solve a sparse and symmetric positive-definite matrix, which can be obtained from the modified nodal analysis. In general, the complexity of direct matrix solve is $O(n^3)$, where n is the number of unknowns. Obviously, with the exponential increase in PDN size with unknowns up to billions, even one direct matrix solve is very time- and resource- consuming. Thus, both academia and industry have spent significant efforts to reduce the simulation cost of PDN sign off [3], [4], [20]–[23].

Recently, with the popularity of machine learning, there were a few works that deployed various machine learning techniques to accelerate the computation of $L\frac{di}{dt}$ noise and *IR* drop without actually invoking the sparse linear system simulator [24]. For example, the work in [25] utilized deep neural network (DNN) models, e.g., fully connected network (FCN) and convolutional neural network (CNN), to predict on-chip supply noise. In [18], the CNN model explicitly used power map, layout, and power bump pattern as input features to compute the full-chip static IR drop. The XGBoost model was also considered as a promising alternative for static IR drop prediction using structural and electrical features extracted from PDN [13], [14], [17], [26]. On the other hand, the dynamic noise can be more challenging as it needs to incorporate more fine-grained local information, e.g., decap, to compute the response to a time-varying input [5], [27]–[29]. Thus, the network models have to rely on the regional local model by dividing the global grids to smaller clips and compute the dynamic noise of a clip within a particular timing window [5], [27]. In addition to the analysis speed-up, there are also a few papers that apply machine learning to help power grid design and optimization [24], [30]–[34]. However, many of the prior works have to incorporate design-dependent features such as location and timing information of cells into power maps and hence make the solution dedicated to a particular design.

B. Post-Layout Circuit Simulation

SPICE-like transistor-level circuit simulation plays an important role in verifying the core design indicators such as accuracy, delay, power consumption, etc. Especially under advanced processes, post-layout circuit simulation has become one of the most time-consuming parts in the design flow [35]. Given the exponentially increasing process complexity and design integration density, machine learning, esp., deep learning, has been considered as a promising alternative to speed up circuit simulation [10], [15], [36].

Unlike the PDN sign off, which deals with one linear system, post-layout circuit simulation needs to account for both linear components from the parasitics and nonlinear components from the transistors. The main task of post-layout circuit simulation is to solve differential algebraic equations (DAEs) established from the modified nodal analysis.Implicit integration methods are then used to convert the differential equations into difference equations [7]. After numerical discretization, a system of nonlinear algebraic equations needs to be solved to obtain circuit state quantity at each time point. Newton Raphson (NR) is a powerful approach to find solutions of these nonlinear equations through linear approximation due to its quadratic convergence property [37]. Therefore, the main simulation problem is transferred to solve a series of linear equations $Ax = b$ at each NR iteration point. The computational bottleneck of the entire simulation is the solution of the Newton equation in the above iterative process, which is to solve a series of large-scale sparse linear systems. In the post-layout simulation, due to the increasingly serious parasitic effects, the execution of sparse linear solvers may account for 60-90% of the total simulation time [38]. Thus, it is highly desired to develop machine learning techniques to accurately predict and control the requested time-step size so as to reduce the number of NR iterations [8] and accelerate the sparse LU factorization [9] at each iteration, e.g. provide selection of reordering algorithms to reduce fill-ins, etc.

Figure 2 shows the flow of transient analysis for post-layout circuit simulation. Before solving the equations, RC reduction [39] and circuit partitioning [40] are the two powerful techniques to decrease the order of linear system to be solved. Hypergraph partitioning, such as patoh [41], hmetis [42], et al., is commonly used to partition a large-scale circuit into small- or medium-scale sub-circuits to facilitate parallel simulation [8]. However, it is difficult to guarantee the load balance for

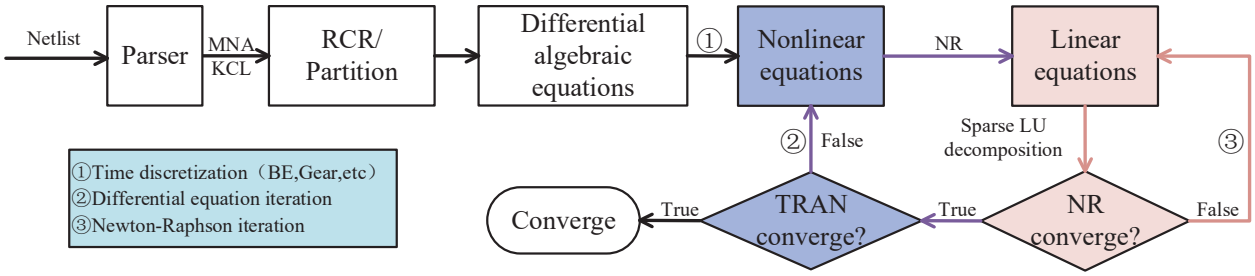

Fig. 2. Transient analysis flow for post-layout transistor level circuit simulation.

each thread during the parallel simulation. Moreover, the components that may induce matrix singularity are usually placed in the same sub-circuit, causing large coupling complexity and hence low parallelism efficiency. Machine learning techniques can then be utilized to enable more efficient circuit partitioning and load balancing [43], [44].

A DC solve is always needed to provide an initial solution to speed up the following transient simulation [45]. There are several popular numerical iterative algorithms to solve the system of nonlinear algebraic equations, including the basic Newton-Raphson (NR) method and continuation methods like Gmin stepping, source stepping, pseudo-transient analysis (PTA), etc [46], [47]. The continuation methods are supposed to be slower in speed but more robust than the plain NR. Its convergence performance are determined by: (i) How to form a continuation function; (ii) How to trace the solution curve; and (iii) How to determine the initial solution [48]. Domain experiences are then highly required to select the appropriate parameters, where machine learning may help replace the domain experiences and offer adaptive continuation iteration for better convergence [15], [47].

III. Application of Deep Learning in Back-End Simulation

Though there are quite a few differences between linear and nonlinear back-end simulations as reviewed in the last section, they actually share similar bottlenecks when deploying machine learning or deep learning techniques for speed up. We can roughly divide the deployment of deep learning to three stages: (i) training; (ii) model architecture; and (iii) model generality. In the following, we will discuss the challenges of deploying deep learning and the corresponding opportunities for future work.

A. Training

The very first stage of any deep learning or machine learning technique deployment is to decide the training dataset and strategy. Most published deep learning techniques in back-end simulation utilize a supervised strategy, i.e., deep learning models are trained on a labelled dataset. The inference accuracy and model generality then largely depends on the quality of annotation and the diversity of training data. In addition, while a more complicated neural network architectures typically yield to more accurate prediction, its required training set size can be also significantly increased to ensure the model

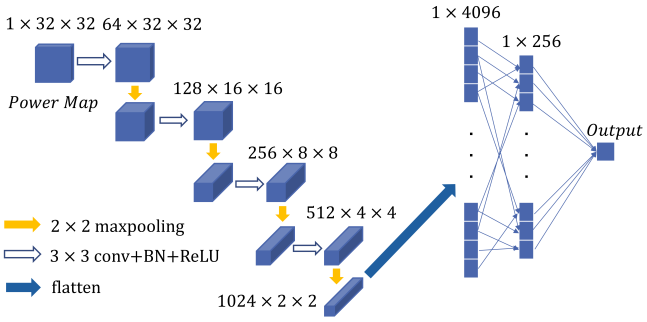

Fig. 3. Network architecture deployed in our experiment.

convergence. However, for back-end design, design houses are usually unwilling to share such training data due to the IP and privacy concerns. Then designers have to go through repeated time-consuming simulations to collect sufficient training data, which can only be used for one particular design. Moreover, even if the quantity of data is resolved, we still need to resolve the following issues to ensure the quality of the data:

- **Insufficient labeled data** It can be even more time-consuming than the simulation to analyze and assign the corresponding labels to each unlabeled data, which demands both expertise and time.
- **Data imbalance** For back-end simulation, designers are more concerned with the simulation results under the worst case, e.g., worst case IR drop, which is however very rare to invoke compared to the other regular cases. Similar observations exist in post-layout circuit simulation. When using deep learning to adaptively decide the proper time step, most training data are collected from the converged phase with larger time steps. However, the samples representing the smaller time steps in the search phase are more critical to the model but happen to be very limited. Thus, the training data appears to be imbalanced and possibly result in misleading inference.

In the following we present an experiment on how the data imbalance may affect the inference accuracy. Here we follow a similar strategy in [5] and design the network architecture in Fig. 3 to classify whether static IR drop exceeds the predefined threshold. Our experiment uses with the 2D design-independent power map as the input and decides the hotspots in the power grid. The case with hotspot is considered as positive sample in the training set whereas the negative sample refers to the IR drop lower than the threshold. Then, by

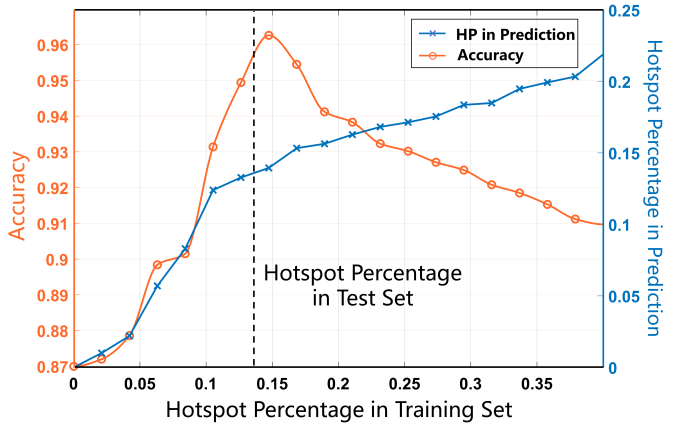

Fig. 4. Impact of data imbalance on inference accuracy and model prediction behavior.

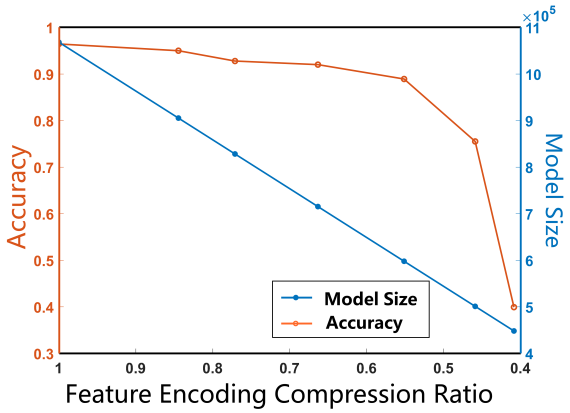

Fig. 5. Impact of feature extraction on prediction accuracy and model complexity

maintaining the same positive sample (or hotspot) percentage in the test set, we change the ratio between the positive and negative samples in the training set and evaluates its impact on the inference. As shown in Fig. 4, the model tends to report more positive cases with an increased positive rate in the training set, while the inference accuracy is actually decreased after certain knee point. Thus, when setting up the training set, it is actually important to balance the positive and negative samples to ensure a more robust model.

B. Model Architecture

Feature extraction is a crucial part of neural network model architecture. A well-designed model only contains the necessary information to reduce the unnecessary computational overhead through feature extraction and selection. Thanks to the huge dimension of the netlists in back-end simulation, the input to the deep learning model can be huge, e.g., millions to billions, which incurs significant memory and time consumption. It is then essential to control the input dimension and only select the necessary features. Many prior works rely on the manually selected features to reduce the input size, which are always design-dependent and demand domain knowledge [13], [14], [17], [26]. Recently, there are also a few works that combine the domain knowledge with encoding schemes to more adaptively select the desired features. For example, to reduce the input dimension while maintaining the essential information, PDN sign off can decompose the spatial and temporal information as a pre-processing scheme [5], [12], [16]. The underlying PDN layout can be spatially divided into clips, the power map of which can then be further encoded to reduce the redundant information. Zhou, et al., explores various feature encoding schemes to reduce the dimension of both input switching and circuit netlist [16]. As shown in Fig. 6, the input switching can be modelled using 1D or 2D encoding to capture the register switching trace, while the netlist can be modelled using graph based encoding. Fig. 5 presents the trade-off between the inference accuracy on PDN noise prediction and model complexity through different feature extractions. It is observed that, even with 36% model complexity reduction, we can still maintain very accurate prediction with only 4% change. Thus, proper feature extraction is critical to the efficiency of the deep learning model, but always remain a challenging problem for different back-end simulations.

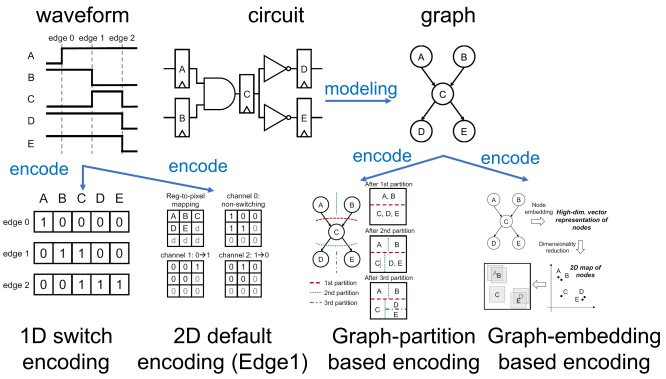

Fig. 6. Encoding schemes for input switching and netlist used in Primal [16] (The Figure is from [16]).

C. Model Generality

The last challenge is the generality of the trained deep learning model. A robust deep learning model can be transferred between different designs or even technologies, which is crucial to the deployment of deep learning in the back-end simulation. However, it is common in prior works that the learned model is dedicated to a particular set of inputs, design and technology. If we categorized the learned features from the training set as specific and general features [49], where the specific features are design-dependent and the general features are universal, it is important to ensure the learned model incorporate more general features than the specific features. Researchers have spent various efforts improving the learning of general features in back-end simulation. For example, GRANNITE [50] adopted a graph neural network (GNN) architecture to learn the general features from both RTL simulation trace and input netlist, which is abstracted as a graph to mitigate the design dependency. Chhabria, et al., proposed a UNet-like structure to learn the universal relationship between the selected features and IR Drops, instead of directly predicting IR drop [18]. For sparse matrix

TABLE I
COMPARISON ON MODEL GENERALITY FOR DIFFERENT DESIGNS AND SCENARIOS

	D1			D2		
	S1	**S2**	**S3**	**S1**	**S2**	**S3**
Toggle Rate	20%	15%	25%	20%	15%	25%
Accuracy	96.46%	92.74%	93.29%	68.06%	66.85%	64.28%

computation, e.g. sparse LU factorization, sparse matrix-vector multiplication (SpMV), etc., that are used in both PDN sign off and post-layout circuit simulation [9], [38], [51], references [52]–[55] suggested machine learning methods to select the proper storage format and implementation kernels. While the model is designed for the underlying mathematical formulation without design details, it is not a trivial task to collect sufficient data and train such a model applicable to all the circuits.

Since the inherent characteristics of industrial-scale circuits are very complicated, the above techniques are more effective when the design is with slight changes. We here use the same IR drop prediction task and the network architecture in Fig. 3 to demonstrate the problem of model generality. Table I summarizes the results, where D1 and D2 refer to two different power grid designs but supplying power to the same circuit netlist; S1, S2 and S3 refer to three different scenarios or test vectors to the circuit. The same test vector to the same circuit may yield to the same excitation. However, thanks to different power grid designs, the reported noise violations are different from design to design and scenario to scenario. The original model is trained on D1/S1 with 96.46% accuracy. While the model can maintain good accuracy across different scenarios (S1, S2, S3), the accuracy significantly drops when transferred to the new power grid design (with the same circuit netlist).

IV. DISCUSSIONS AND CONCLUSIONS

In this paper, we discussed a few existing challenges of deploying machine learning or deep learning models in back-end simulation. While it is observed growing interest in deep learning assisted back-end simulation, there are still many practical issues that need to be addressed, as discussed in the last section. Even with all the above issues, the popularity of deep learning still open up many opportunities to the back-end simulation area. For example, federated learning can be a promising alternative to unify the data from different design houses to resolve the issue of data insufficiency and homogeneity while maintaining local data privacy [56]. While many models demand very particular domain knowledge or heuristics to design the proper model architecture, GNN can be a natural option to properly model the circuit and input stimulus, and then extract the intrinsic features of complex circuits [57]. For the model generality, transfer learning has been proposed to increase the generality, in which the features in shallow layers can be adapted to other tasks [58].

However, despite all the new techniques, it is noted that the back-end simulation is well formulated in mathematics. Very few machine learning works attempt to theoretically assist the underlying mathematical formulation and matrix solve, as the overhead of which can be non-trivial in comparison to the other speed-up techniques, such as parallel execution. On the other hand, it seems more promising to apply deep learning to the design or metric modeling so as to reduce the back-end problem complexity, which demands domain knowledge to achieve an efficient model. Moreover, since back-end simulation plays a very critical role in chip sign off, the model fidelity is important to provide designers with high confidence in deploying the technique.

ACKNOWLEDGMENT

This work was supported in part by Zhejiang Provincial Key R&D program (Grant No. 2020C01052), NSFC (Grant No. 62034007 and 61974133) and Science Foundation of China University of Petroleum, Beijing (No. 2462020YXZZ024).

REFERENCES

[1] C. K. Sarkar, *Technology computer aided design: simulation for VLSI MOSFET*. CRC Press, 2013.

[2] B. Khailany, H. Ren, S. Dai, S. Godil, B. Keller, R. Kirby, A. Klinefelter, R. Venkatesan, Y. Zhang, B. Catanzaro, and W. J. Dally, "Accelerating chip design with machine learning," *IEEE Micro*, vol. 40, no. 6, pp. 23–32, 2020.

[3] C. Zhuo, J. Hu, M. Zhao, and K. Chen, "Power grid analysis and optimization using algebraic multigrid," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 738–751, 2008.

[4] C. Zhuo, G. Wilke, R. Chakraborty, A. A. Aydiner, S. Chakravarty, and W.-K. Shih, "Silicon-validated power delivery modeling and analysis on a 32-nm ddr i/o interface," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 23, pp. 1760–1771, 2015.

[5] Z. Xie, H. Ren, B. Khailany, Y. Sheng, S. Santosh, J. Hu, and Y. Chen, "Powernet: Transferable dynamic ir drop estimation via maximum convolutional neural network," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 13–18, IEEE, 2020.

[6] I. N. Hajj, "Circuit theory in circuit simulation," *IEEE Circuits and Systems Magazine*, vol. 16, no. 2, pp. 6–10, 2016.

[7] Q. Chen, "A robust exponential integrator method for generic nonlinear circuit simulation," in *in Proceedings of the 57th Annual Design Automation Conference*, pp. 1–6, IEEE, 2020.

[8] Z. Jin, T. Feng, Y. Duan, X. Wu, M. Cheng, Z. Zhou, and W. Liu, "Palbbd: A parallel arclength method using bordered block diagonal form for dc analysis," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pp. 327–332, 2021.

[9] J. Zhao, Y. Wen, Y. Luo, Z. Jin, W. Liu, and Z. Zhou, "Sflu: Synchronization-free sparse lu factorization for fast circuit simulation on gpus," *in Proceedings of the 58th Annual Design Automation Conference, IEEE*, pp. 37–42, 2021.

[10] Q. Zhang, S. Su, J. Liu, and M. S.-W. Chen, "Cepa: Cnn-based early performance assertion scheme for analog and mixed-signal circuit simulation," in *in Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, IEEE, 2020.

[11] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[12] V. A. Chhabria, Y. Zhang, H. Ren, B. Keller, B. Khailany, and S. S. Sapatnekar, "Mavirec: Ml-aided vectored ir-drop estimation and classification," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1825–1828, IEEE, 2021.

[13] C.-H. Pao, A.-Y. Su, and Y.-M. Lee, "Xgbir: an xgboost-based ir drop predictor for power delivery network," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1307–1310, IEEE, 2020.

[14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[15] W. W. Xing, X. Jin, Y. Liu, D. Niu, W. Zhao, and Z. Jin, "Boa-pta, a bayesian optimization accelerated error-free spice solver," *arXiv preprint arXiv:2108.00257*, 2021.

[16] Y. Zhou, H. Ren, Y. Zhang, B. Keller, B. Khailany, and Z. Zhang, "Primal: Power inference using machine learning," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

[17] C.-T. Ho and A. B. Kahng, "Incpird: Fast learning-based prediction of incremental ir drop," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2019.

[18] V. A. Chhabria, V. Ahuja, A. Prabhu, N. Patil, P. Jain, and S. S. Sapatnekar, "Thermal and ir drop analysis using convolutional encoder-decoder networks," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pp. 690–696, 2021.

[19] W. Yu, Z. Xu, B. Li, and C. Zhuo, "Floating random walk-based capacitance simulation considering general floating metals," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1711–1715, 2018.

[20] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 2, pp. 159–168, 2002.

[21] Z. Zhu, B. Yao, and C.-K. Cheng, "Power network analysis using an adaptive algebraic multigrid approach," in *Proceedings of the 40th annual Design Automation Conference*, pp. 105–108, 2003.

[22] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 8, pp. 1204–1224, 2005.

[23] C. Zhang and P. Zhou, "Improved hierarchical ir drop analysis in homogeneous circuits," in *2020 IEEE 15th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*, pp. 1–3, IEEE, 2020.

[24] C. Zhuo, K. Unda, Y. Shi, and W.-K. Shih, "From layout to system: Early stage power delivery and architecture co-exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 7, pp. 1291–1304, 2019.

[25] S. N. Mozaffari, B. Bhaskaran, K. Narayanun, A. Abdollahian, V. Pagalone, S. Sarangi, and J. E. Colburn, "An efficient supervised learning method to predict power supply noise during at-speed test," in *2019 IEEE International Test Conference (ITC)*, pp. 1–10, IEEE, 2019.

[26] S.-Y. Lin, Y.-C. Fang, Y.-C. Li, Y.-C. Liu, T.-S. Yang, S.-C. Lin, C.-M. Li, and E. J.-W. Fang, "Ir drop prediction of eco-revised circuits using machine learning," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, pp. 1–6, IEEE, 2018.

[27] Y. Kwon, G. Jung, D. Hyun, and Y. Shin, "Dynamic ir drop prediction using image-to-image translation neural network," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2021.

[28] Y.-C. Fang, H.-Y. Lin, M.-Y. Su, C.-M. Li, and E. J.-W. Fang, "Machine-learning-based dynamic ir drop prediction for eco," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 1–7, 2018.

[29] Y. Li, C. Zhuo, and P. Zhou, "A cross-layer framework for temporal power and supply noise prediction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1914–1927, 2019.

[30] H. Zhou, W. Jin, and S. X.-D. Tan, "Gridnet: Fast data-driven em-induced ir drop prediction and localized fixing for on-chip power grid networks," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2020.

[31] V. A. Chhabria, A. B. Kahng, M. Kim, U. Mallappa, S. S. Sapatnekar, and B. Xu, "Template-based pdn synthesis in floorplan and placement using classifier and cnn techniques," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 44–49, IEEE, 2020.

[32] S. Dey, S. Nandi, and G. Trivedi, "Powerplanningdl: Reliability-aware framework for on-chip power grid design using deep learning," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1520–1525, IEEE, 2020.

[33] X.-X. Huang, H.-C. Chen, S.-W. Wang, I. H.-R. Jiang, Y.-C. Chou, and C.-H. Tsai, "Dynamic ir-drop eco optimization by cell movement with current waveform staggering and machine learning guidance," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

[34] H.-Y. Lin, Y.-C. Fang, S.-T. Liu, J.-X. Chen, C.-M. Li, and E. J.-W. Fang, "Automatic ir-drop eco using machine learning," in *2020 IEEE International Test Conference in Asia (ITC-Asia)*, pp. 7–12, IEEE, 2020.

[35] C. Zhao, Z. Zhou, and D. Wu, "Empyrean ALPS-GT: gpu-accelerated analog circuit simulation," in *IEEE/ACM International Conference On Computer Aided Design, (ICCAD)*, pp. 167:1–167:3, IEEE, 2020.

[36] J. Li, M. Yue, Y. Zhao, and G. Lin, "Machine-learning-based online transient analysis via iterative computation of generator dynamics," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2020.

[37] T. Nakura, *SPICE Simulation*, pp. 19–47. Singapore: Springer, 2016.

[38] S. Peng and S. X. Tan, "GLU3.0: fast gpu-based parallel sparse LU factorization for circuit simulation," *IEEE Des. Test*, vol. 37, no. 3, pp. 78–90, 2020.

[39] P. Benner, M. Hinze, and E. J. W. Ter Maten, *Model reduction for circuit simulation*, vol. 74. Springer, 2011.

[40] F. M. Johannes, "Partitioning of vlsi circuits and systems," in *Proceedings of the 33rd Annual Design Automation Conference*, pp. 83–87, 1996.

[41] Ü. V. Çatalyürek and C. Aykanat, "Patoh (partitioning tool for hypergraphs)," in *Encyclopedia of Parallel Computing*, pp. 1479–1487, Springer, 2011.

[42] G. Karypis, "hmetis 1.5: A hypergraph partitioning package," *http://www. cs. umn. edu/˜ metis*, 1998.

[43] F. bizzarri, A. Brambilla, and G. Storti-Gajani, "Fastspice circuit partitioning to compute dc operating points preserving spice-like simulators accuracy," *Simulation Modelling Practice and Theory*, vol. 81, pp. 51–63, 2018.

[44] N. Zhu, "Partitioning in post-layout circuit simulation," January 2019.

[45] F. N. Najm, *Circuit simulation*. John Wiley & Sons, 2010.

[46] T. Najibi, "Continuation methods as applied to circuit simulation," *IEEE Circuits and Devices Magazine*, vol. 5, no. 5, pp. 48–49, 1989.

[47] Z. JIN, M. LIU, and X. WU, "An adaptive dynamic-element pta method for solving nonlinear dc operating point of transistor circuits," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 37–40, 2018.

[48] A. Ushida, Y. Yamagami, Y. Nishio, I. Kinouchi, and Y. Inoue, "An efficient algorithm for finding multiple dc solutions based on the spice-oriented newton homotopy method," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 21, no. 3, pp. 337–348, 2002.

[49] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3320–3328, 2014.

[50] Y. Zhang, H. Ren, and B. Khailany, "Grannite: Graph neural network inference for transferable power estimation," in *in Proceedings of the 57th Annual Design Automation Conference*, IEEE, 2020.

[51] W. Lee and R. Achar, "Gpu-accelerated adaptive PCBSO mode-based hybrid RLA for sparse LU factorization in circuit simulation," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2320–2330, 2021.

[52] E. Dufrechou, P. Ezzatti, and E. S. Quintana-Ortí, "Selecting optimal spmv realizations for gpus via machine learning," *Int. J. High Perform. Comput. Appl.*, vol. 35, no. 3, 2021.

[53] H. Cui, S. Hirasawa, H. Kobayashi, and H. Takizawa, "A machine learning-based approach for selecting spmv kernels and matrix storage formats," *IEICE Trans. Inf. Syst.*, vol. 101-D, no. 9, pp. 2307–2314, 2018.

[54] I. Nisa, C. Siegel, A. Sukumaran-Rajam, A. Vishnu, and P. Sadayappan, "Effective machine learning based format selection and performance modeling for spmv on gpus," in *IEEE International Parallel and Distributed Processing Symposium Workshops*, pp. 1056–1065, 2018.

[55] R. Furuhata, M. Zhao, M. Agung, R. Egawa, and H. Takizawa, "Improving the accuracy in spmv implementation selection with machine learning," in *Eighth International Symposium on Computing and Networking Workshops (CANDAR)*, pp. 172–177, IEEE, 2020.

[56] X. Lin, J. Pan, J. Xu, Y. Chen, and C. Zhuo, "Lithography hotspot detection via heterogeneous federated learning with local adaptation," *arXiv preprint arXiv:2107.04367*, 2021.

[57] 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 *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

[58] C. Yu and W. Zhou, "Decision making in synthesis cross technologies using lstms and transfer learning," in *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, MLCAD '20, pp. 55–60, Association for Computing Machinery, 2020.