WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Machine Learning for VLSI Physical Design and EDA Optimization: A Literature Review

Emandi Bilva[1], Gollu Venkata Surya Narayana[2], Rajapantula Kranthi[3]

[1,2,3] Department of ECE, GMRIT, Deemed to be University, Rajam, Vizianagaram, Andhra Pradesh-532127.
Emails: 1bilvaemandi@gmail.com, 2kranthi.rajapantula@gmail.com, 3gollusurya7326@gmail.com

ABSTRACT

Machine learning (ML) is increasingly applied to VLSI physical design to predict down-stream implementation outcomes and guide design decisions. This review examines research on floorplanning and placement, routing and routability, and optimization across the design flow. Although studies report promising results in GPU-accelerated placement, learned congestion guidance, and reinforcement-learning (RL) methods, the available evidence does not establish consistent superiority over strong alternatives across unseen designs or sign-off conditions. Current evidence supports the use of ML as an adjunct to established electronic design automation (EDA) tools, particularly for guiding search and reducing expensive evaluations; generalization, reproducibility, and post-route performance remain unresolved concerns.

Keywords: VLSI; physical design; electronic design automation; machine learning; place-ment; floorplanning; routability; reinforcement learning; congestion prediction

1. Introduction

Turning a logical netlist into a physical layout involves several linked steps, including floorplanning, placement, clock-tree synthesis, routing, and sign-off. Choices made early in the flow can shape congestion and wirelength later on; routing and timing results, in turn, may force earlier decisions to be revisited. With so many possible configurations and repeated tool runs, physical design offers natural opportunities for prediction and optimization. [1–3] [4]

In this literature, ML plays three related but distinct roles. Some models predict a later result such as congestion, timing, routability, or power integrity from information available earlier in the flow. Others choose or tune design actions, using approaches such as supervised learning, search, or reinforcement learning. A third group uses deep-learning software and GPUs to speed up computation, even when the optimizer itself remains analytical. Keeping these roles separate helps make sense of reported results: faster computation is not the same as a learned policy finding a better design, and an accurate prediction matters only if it helps improve a real tool decision. [5]

2. Scope and approach

This is a focused narrative review rather than a protocol-registered systematic review or meta-analysis. It draws on surveys and empirical studies of ML in physical design, with emphasis on placement and floorplanning, routability prediction, routing, and automated tool tuning. Studies were most useful when they reported benchmark results, connected a model to a physical-design flow, or examined generalization and reproducibility. Because the studies use different designs, tools, objectives, and baselines, their reported metrics should not be treated as a direct ranking. Some papers were available only as abstracts; accordingly, the discussion gives greatest weight to sources with accessible article text or concrete results in the abstract. [4, 5]

3. Machine learning fits in physical design

3.1 Placement and floorplanning

Placement and floorplanning determine where circuit components and macros sit on the chip. The choices must balance wirelength, density, timing, and routability. Graph-based models offer a way to represent netlist connectivity, while RL treats placement as a sequence of decisions. Other methods leave the analytical placer in charge and use ML to predict outcomes or tune its settings. [4, 6, 7]

One influential RL approach trains an agent to place netlist nodes on a chip canvas. The method uses learned representations of placement quality to help the policy transfer experience between blocks. The authors report generating placements for modern accelerator netlists in under six hours, compared with weeks of expert effort for the baseline process they describe. [7] This result is specific to the workloads and evaluation conditions reported and does not establish that RL outperforms conventional placement across a broader design population.

GraphPlanner uses a graph-convolutional model to learn the relationship between circuit con-nectivity and wirelength, then combines it with mixed-size placers. In the reported academic and industrial evaluations, the authors found an average 25% reduction in placement runtime and a 4% reduction in wirelength compared with the placers alone. [6] AutoDMP takes a more search-oriented approach, combining GPU-accelerated DREAMPlace with multi-objective hyperparameter optimization for macro placement. Its authors report optimizing a design containing 2.7 million cells and 320 macros in three hours on a single DGX Station A100. [8] Collectively, these studies support hybrid architectures in which ML guides or tunes established implementation engines rather than replacing the full design flow.

3.2 Routing and routability prediction

Routing is costly, so designers would benefit from knowing earlier which placements are likely to create trouble. One study added a convolutional congestion predictor to DREAMPlace and used its predicted hotspots to guide placement. On ISPD 2015 benchmarks, the authors report up to 9.05% lower congestion rate and 5.30% lower routed wirelength than their comparison baseline. [2] The evaluation is informative because the predictions guided placement and were assessed against routing outcomes. Standalone prediction accuracy would not establish improvement in the design flow.

Other studies estimate congestion or detailed-route violations from placed designs, or use learned heatmaps to guide global routing. A 2025 multi-task method, for example, predicts congestion and design-rule-check violations together, using shared features to model the two tasks jointly. [4, 9] These methods face two related challenges: downstream labels can be costly to generate, and model performance may not transfer across designs, tools, or technologies. Random train-test partitions within a single benchmark are insufficient to establish such transferability.

3.3 GPU acceleration and analytical placement

DREAMPlace shows how deep-learning infrastructure can accelerate a conventional optimization method. It reformulates analytical placement to use deep-learning software and custom GPU kernels; the paper reports roughly a 40-fold speedup in global placement without loss of quality relative to the cited multithreaded placer. [1] This contribution concerns computational acceleration rather than a learned placement policy. Although faster placement may enable broader search, it does not by itself demonstrate improved objective quality attributable to neural-network or RL decision-making.

3.4 Optimization across the design flow

ML can also look beyond a single placement step. Models can estimate downstream tool results, tune settings, or help decide which flow runs are worth continuing. Physical-design research has long considered prediction of timing correlation, routability, and later implementation outcomes as a way to reduce unnecessary margins and repeated runs. [10] Broader EDA surveys likewise describe ML as a way to navigate design and configuration spaces that are too large for exhaustive search. [5]. A principal risk is ranking error when a surrogate is applied to a tool version, design family, or operating regime that differs from its training distribution. Accordingly, predictions should be accompanied by uncertainty estimates, fallback procedures, and validation against implementation or sign-off tools. Prior work also emphasizes the importance of modeling both design characteristics and tool behavior. [5, 10]

4. Preliminaries

Reported gains are study-specific and should be interpreted in light of differences in benchmarks, objectives, toolchains, and comparison methods. Consequently, published headline values do not support direct ranking across methods. Improvements in wirelength do not necessarily imply better routed timing or lower power, and congestion prediction may benefit one stage without optimizing all power-performance-area objectives. [2, 6, 8]

RL macro placement shows why independent evaluation matters. The original work reports learning from previous placements and generating solutions for blocks not seen during training. [7] A later assessment compared the approach with stronger simulated-annealing baselines, added public sub-10-nm test cases, and evaluated post-route power, performance, and area. It also identified unanswered questions about scalability, pre-training, and reproducibility. [3] Taken together, these findings support RL placement as a promising but context-dependent approach; they do not establish it as a general replacement for conventional optimization.

The validity of the training target is as important as model selection. An EDA survey describes a model predicting design-rule violations for macro placements containing fewer than six macros, without incorporating subsequent place-and-route optimization. Because implementation may alter both placement and the resulting violations, the predicted values may not represent final outcomes. [4] More generally, a proxy is useful only if it preserves the ranking of decision-relevant final outcomes. Otherwise, a model may perform well on its training metric while directing the design flow toward inferior implementations.

5. Priorities for future work

- **Test transfer between designs and flows.** Hold out entire designs, sizes, and design families—not just random samples from the same benchmark. Report whether fine-tuning is needed and how much new tool data it takes.
- **Measure what matters after routing.** Where relevant, report routed wirelength, timing, power, congestion, design-rule violations, and legality. Make clear when a gain is only in a proxy metric rather than in the final implementation.
- **Use strong, reproducible comparisons.** Compare against tuned classical heuristics, established open-source tools, and commercial flows where possible. Share benchmark definitions, constraints, tool settings, seeds, and evaluation scripts.
- **Count the full cost.** Include the effort and compute needed to generate labels and train models, as well as inference time, tool-license usage, and end-to-end runtime. A fast prediction may not save time if data generation or failure recovery dominates.
- **Handle uncertainty and distribution shift.** Use uncertainty to decide when a model can guide a run and when the flow should fall back to authoritative tools. Recheck performance as designs and tool versions change.
- **Treat design quality as multi-objective.** Evaluate trade-offs among area, performance, power, routability, and reliability rather than using one metric as a stand-in for overall quality. [3–5]

6. Conclusion

ML has demonstrated potential in VLSI physical design, particularly for routability guidance, netlist-aware prediction, tool tuning, and GPU acceleration. The strongest current case is for hybrid approaches in which ML narrows the search or predicts costly downstream outcomes, while established EDA tools perform implementation and validation. The available evidence does not support a general claim of superiority over strong alternatives. Such a conclusion requires independent evaluation across diverse designs using post-route and sign-off outcomes, rather than results from a single benchmark. [2, 3, 8]

7. References

1. Y. Lin, Z.-X. Jiang, J. Gu, et al. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020. https://doi.org/10.1109/TCAD.2020.3003843
2. S. Liu, Q. Sun, P.-Y. Liao, et al. Global Placement with Deep Learning-Enabled Explicit Routability Optimization. In *Design, Automation and Test in Europe (DATE)*, 2021. https://doi.org/10.23919/DATE51398.2021.9473959
3. C.-K. Cheng, A. Kahng, S. Kundu, et al. An Updated Assessment of Reinforcement Learning for Macro Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023. https://doi.org/10.1109/TCAD.2025.3644293
4. K. I. Gubbi, S. A. Beheshti-Shirazi, T. Sheaves, et al. Survey of Machine Learning for Electronic Design Automation. In *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2022. https://doi.org/10.1145/3526241.3530834
5. M. Rapp, H. Amrouch, Y. Lin, et al. MLCAD: A Survey of Research in Machine Learning for CAD (Keynote Paper). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022. https://doi.org/10.1109/TCAD.2021.3124762
6. Y. Liu, Z. Ju, Z. Li, et al. GraphPlanner: Floorplanning with Graph Neural Network. *ACM Transactions on Design Automation of Electronic Systems*, 2022. https://doi.org/10. 1145/3555804
7. Mirhoseini, A. Goldie, M. Yazgan, et al. Chip Placement with Deep Re-inforcement Learning. arXiv, 2020. https://www.semanticscholar.org/paper/ 929bf1a2ff229d34f7907886989c621444c2b8fd
8. A. Agnesina, P. Rajvanshi, T. Yang, et al. AutoDMP: Automated DREAMPlace-based Macro Placement. In *ACM International Symposium on Physical Design (ISPD)*, 2023. https://doi.org/10.1145/3569052.3578923
9. J.-Q. Chen and G.-G. Liu. Multi-Task Learning for Routability Prediction. In *2025 IEEE 5th International Conference on Software Engineering and Artificial Intelligence (SEAI)*, 2025. https://doi.org/10.1109/SEAI65851.2025.11108727
10. A. Kahng. Machine Learning Applications in Physical Design: Recent Results and Directions. In *ACM International Symposium on Physical Design (ISPD)*, 2018. https://doi.org/ 10.1145/3177540.3177554