MANTECH Publications

Journal of Advanced VLSI Architectures and Nanoelectronics
Volume 2, Issue 1, January-April 2026
ISSN: 3108-2351 (Online)

AI/ML Assisted VLSI Design Automation

Rajesh Kumar Pathak[1], Kuldeep Singh[2], Suraj S Tiwari[3]

Lecturer[1], PG Scholar[2, 3]

Department of ECE

The Rashtrasant Tukadoji Maharaj Nagpur University

***Corresponding Author Email**: Rajeshkumarpathak41@yahoo.com[1]*

***DOI:** https://doi.org/10.5281/zenodo.19816981*

ABSTRACT

The VLSI (Very Large Scale Integration) design process has become increasingly complex due to the scaling of technology nodes and the rising demand for high-performance, low-power integrated circuits. Traditional Electronic Design Automation (EDA) tools often face challenges in achieving optimal design solutions efficiently. Artificial Intelligence (AI) and Machine Learning (ML) techniques are emerging as promising approaches to enhance VLSI design automation by improving tasks such as logic synthesis, placement, routing, and verification. This review paper discusses recent advancements in AI/ML-assisted VLSI design, highlighting the applications of supervised learning, reinforcement learning, and deep learning in different stages of the design flow. Various case studies and comparative analyses of AI-driven methodologies versus conventional techniques are presented. The paper also explores future directions and challenges in integrating AI/ML into mainstream VLSI design workflows.

***KEYWORDS:** VLSI, AI, ML, EDA, Placement, Routing, Logic Synthesis, Verification, Neural Networks, Reinforcement Learning*

INTRODUCTION

VLSI technology has revolutionized modern electronics, enabling billions of transistors to be integrated onto a single chip. As process nodes shrink and design complexity grows, the traditional design cycle becomes increasingly time-consuming and resource-intensive.

Conventional EDA tools rely heavily on heuristics and human expertise to manage tasks like placement, routing, and timing optimization.

The integration of AI and ML into VLSI design automation offers a paradigm shift. ML models can learn from large design datasets, predict optimal solutions, and adaptively improve performance without exhaustive manual tuning. AI techniques, including deep learning (DL) and reinforcement learning (RL), are increasingly being explored to tackle NP-hard problems in VLSI design. This paper reviews AI/ML methods applied to VLSI design automation, highlighting their benefits, limitations, and future potential.

BACKGROUND AND MOTIVATION

The increasing complexity of modern integrated circuits has made VLSI design a highly intricate process that demands sophisticated automation tools. Traditional EDA (Electronic Design Automation) tools have been instrumental in managing these complexities, but with the advent of advanced technology nodes (e.g., 7nm, 5nm, and below), conventional approaches face several limitations. AI and ML techniques are gaining attention as potential solutions to improve efficiency, accuracy, and scalability in the VLSI design process.

This section elaborates on the typical VLSI design flow and the challenges that motivate the adoption of AI/ML-driven methods.

1. VLSI Design Flow

The VLSI design flow is a multi-stage process that transforms high-level specifications into a physically realizable silicon chip. Each stage presents unique challenges that involve optimization problems, often NP-hard in nature. A brief elaboration of these stages is as follows:

a) Specification and RTL Design:

The design process begins with the collection of functional requirements for the intended chip. Designers use Hardware Description Languages (HDLs), such as Verilog or VHDL, to implement the Register Transfer Level (RTL) representation of the design. The RTL captures the functional behavior of the system without detailing its physical implementation. At this stage, design correctness, functional completeness, and modularity are primary concerns.

AI/ML can assist in automated code generation, RTL error prediction, and early identification of design bottlenecks.

b) Logic Synthesis:

Logic synthesis transforms the RTL design into a gate-level netlist, mapping abstract logic operations into physical gates available in a target standard-cell library. This step involves optimizing for three major design parameters: area, power, and performance. Traditional synthesis relies on rule-based heuristics and pre-defined optimization scripts, which may not be adaptive to complex designs. ML models can predict optimal synthesis strategies based on prior designs, potentially reducing runtime and improving solution quality.

c) Placement:

Placement determines the physical locations of standard cells or modules on the silicon die. The objective is to minimize total interconnect wirelength, satisfy timing constraints, and reduce congestion. Placement is inherently an NP-hard problem because the number of possible placements grows exponentially with the number of cells. AI approaches, such as reinforcement learning and graph neural networks, can intelligently explore the solution space, learning placement patterns that traditional heuristics may overlook.

d) Routing:

Routing involves creating the physical connections (wires) between placed cells. This stage must consider congestion, crosstalk, signal integrity, and manufacturability rules. Routing problems are often NP-hard due to the large number of nets and limited routing resources. AI/ML models can predict congested regions, optimize routing paths, and even dynamically guide routing algorithms to improve efficiency and reduce iterations.

e) Verification and Testing:

Verification ensures the functional and timing correctness of the design. Traditional verification involves extensive simulations, formal verification, and emulation, which are time-consuming and computationally expensive. ML models can assist in prioritizing test cases, predicting potential error-prone regions, and reducing simulation time while maintaining high coverage. AI-assisted verification has the potential to significantly shorten design cycles and improve reliability.

Each of these stages presents optimization challenges where exhaustive search is impractical. AI/ML techniques can leverage historical data, patterns, and heuristics to intelligently guide the design process, making them ideal candidates for integration into modern VLSI flows.

CHALLENGES IN TRADITIONAL EDA TOOLS

Despite significant advancements, traditional EDA tools face several limitations that motivate the adoption of AI/ML methods:

1. Scalability Issues:

With modern designs integrating billions of transistors, the design space has expanded exponentially. Traditional algorithms, which often rely on exhaustive or combinatorial searches, cannot scale efficiently to such large problem sizes. This leads to longer design cycles and increased computational requirements.

2. Sub-optimal Solutions:

Conventional heuristic algorithms, such as simulated annealing or force-directed placement, aim to find well-enough solutions but cannot guarantee global optimality. This is especially problematic for placement, routing, and power optimization, where even small inefficiencies can significantly impact chip performance. AI/ML models, trained on past designs, can generalize patterns to predict near-optimal solutions more effectively.

3. Long Iteration Cycles:

Iterative design improvements in traditional workflows require repeated simulation, placement, and routing steps. Each iteration consumes valuable time and computational resources. AI/ML approaches, particularly predictive models and reinforcement learning agents, can accelerate convergence and reduce the number of iterations needed.

4. Design Rule Complexity:

Advanced process nodes impose stringent design rules for manufacturing, such as double patterning, via constraints, and electromigration limits. Traditional EDA tools struggle to simultaneously optimize for all these rules. AI/ML models can learn correlations between design parameters and rule violations, providing intelligent guidance to designers.

Role of AI/ML in Overcoming Challenges:

AI/ML techniques offer several advantages in addressing these limitations:

- **Learning from Experience:** Historical design data can be used to train models that predict optimal configurations, reducing reliance on exhaustive search.
- **Adaptive Optimization:** Models can dynamically adapt to different design sizes, topologies, and process nodes.
- **Automation of Repetitive Tasks:** Repetitive decision-making, such as gate placement or routing path selection, can be automated, freeing designers to focus on higher-level design decisions.
- **Predictive Analysis:** AI/ML can predict timing, power, congestion, and error-prone areas early in the design flow, reducing costly iterations.

In summary, the increasing complexity of modern VLSI designs, combined with the limitations of traditional EDA tools, creates a strong motivation for incorporating AI and ML techniques into the design automation process. These methods promise enhanced efficiency, accuracy, and scalability, paving the way for faster and more reliable chip design.

AI/ML TECHNIQUES IN VLSI DESIGN AUTOMATION

The integration of AI and ML techniques into VLSI design automation is increasingly important as traditional EDA tools face scalability and efficiency challenges. Various ML methodologies, including supervised learning, reinforcement learning, deep learning, and hybrid approaches, are being applied at multiple stages of the design flow. Each technique offers unique advantages depending on the nature of the problem—predictive, sequential, or combinatorial.

1. Supervised Learning

Supervised learning is one of the most widely used ML approaches in VLSI design automation. In supervised learning, a model is trained on labeled datasets, where input features correspond to design parameters, and labels represent the expected outcome, such as timing, power, or congestion. Once trained, the model can predict outcomes for unseen designs, reducing the need for exhaustive simulations or iterative trial-and-error approaches.

Supervised learning is particularly effective for **parameter prediction** tasks in VLSI, such as timing, power, and routing congestion.

a) Timing Prediction

Timing analysis is a critical step in VLSI design, ensuring that signal paths meet the required setup and hold times. Traditional timing analysis requires simulating the netlist under various conditions, which is time-consuming for large designs.

Supervised learning models, particularly **feedforward neural networks** (FNNs) or **gradient boosting regressors**, can predict timing slack or violations based on input features such as:

- Gate delays and types
- Net fanout and fanin
- Interconnect lengths
- Previous design layouts

The model is trained on a dataset of designs with known timing results. After training, it can quickly predict timing violations for new netlists, significantly reducing simulation time. Studies have shown that neural network-based timing predictors can achieve over **90% accuracy** while reducing analysis time by 60–70%.

b) Power Estimation

Power consumption is a key metric in modern chips, affecting battery life in mobile devices and thermal management in high-performance systems. Power estimation involves both **dynamic power** (switching activity) and **static power** (leakage currents).

Supervised learning models, such as **random forests, support vector regression (SVR), and neural networks**, can estimate power consumption using features like:

- Gate counts and types
- Switching activity and toggling rates
- Placement and routing characteristics
- Voltage and frequency specifications

For instance, a trained random forest regressor can predict the total power of an ASIC design without running full SPICE-level simulations. This allows designers to explore multiple architectural and placement alternatives efficiently.

c) Congestion Prediction

Routing congestion occurs when too many interconnects compete for limited routing resources, which can lead to timing violations or design rule violations. Traditionally, congestion analysis is performed post-placement and post-routing, often requiring multiple iterations to resolve hotspots.

Supervised learning classifiers, such as **support vector machines (SVMs), decision trees, or convolutional neural networks (CNNs)**, can predict congestion during or even before placement. Input features for these models include:

- Cell density in local regions
- Number of nets and netlength estimates
- Connectivity patterns
- Previous design layouts or benchmarks

By predicting congestion hotspots early, the placement algorithm can adjust cell locations proactively, reducing the number of routing iterations and improving overall chip performance.

d) Advantages of Supervised Learning in VLSI

- **Fast Predictions:** Once trained, models provide near-instant predictions for timing, power, and congestion.
- **Design Space Exploration:** Enables evaluation of multiple design alternatives without full simulations.
- **Reduced Iterations:** Predictive insights allow early detection of potential design issues, reducing costly design cycles.
- **Data-Driven Decisions:** Learns from historical design data, capturing patterns that heuristics may miss.

e) Limitations

- **Dependence on Labeled Data:** Requires a substantial dataset of prior designs with accurate labels.
- **Generalization Issues:** Models trained on a specific technology node or design style may not perform well on different designs.
- **Feature Selection Complexity:** Identifying relevant input features for accurate prediction can be challenging.

Table 1: Supervised Learning Applications in VLSI

Task	ML Model	Input	Output	Reference
Timing Analysis	Feedforward Neural Network	Netlist features, gate delays	Timing slack	[1]
Power Estimation	Random Forest	Gate counts, switching activity	Power consumption	[2]
Congestion Prediction	SVM	Placement coordinates, net density	Congestion map	[3]

2. Reinforcement Learning (RL)

Reinforcement Learning (RL) is a branch of machine learning that is well-suited for **sequential decision-making problems**, where an agent interacts with an environment and learns to take actions that maximize a cumulative reward. In VLSI design, many optimization tasks—such as placement, routing, and floorplanning—can be modeled as sequential decision problems, making RL an ideal approach.

a) RL for Placement Optimization

Placement optimization involves determining the physical locations of cells or modules on the chip, with the objective of minimizing wirelength, congestion, and timing violations. The large search space and complex dependencies between cells make this an NP-hard problem.

RL formulates placement as an **environment-agent problem**:

- **Environment:** Represents the chip layout, netlist connectivity, and design rules.
- **Agent:** A neural network or policy model that decides the placement of each cell.
- **State:** Current placement configuration of the chip.
- **Action:** Placement of a particular cell or module in a specific location.
- **Reward:** A scalar value representing design quality, such as wirelength reduction, timing closure, or congestion minimization.

Through trial-and-error exploration and policy updates, the RL agent learns to place cells in a manner that optimizes the overall design metrics. Studies have shown that RL-based placement

can achieve up to **10–15% reduction in wirelength** and improve timing closure compared to conventional heuristics like simulated annealing.

b) RL for Routing Path Selection

Routing is another sequential optimization problem where the objective is to connect all placed cells while minimizing congestion, crosstalk, and delay. Traditional routers often rely on deterministic or heuristic algorithms that can get trapped in local optima.

In RL-based routing:

- The **state** represents the current routing grid, existing connections, and congestion maps.
- The **action** involves choosing the next routing path for a net or segment.
- The **reward** is based on metrics such as reduced congestion, minimized crosstalk, and adherence to design rules.

RL agents can adaptively select routing paths and explore multiple solutions in parallel, enabling faster convergence and more efficient use of routing resources.

c) RL for Floorplanning

Floorplanning determines the macro-level placement of functional blocks on a chip. RL can dynamically optimize floorplans by learning policies that balance **area utilization, wirelength, and timing trade-offs**. The RL agent can experiment with block arrangements, evaluate rewards based on metrics like total area, critical path length, and congestion, and iteratively improve the layout.

d) Advantages of RL in VLSI

- **Sequential Decision Optimization:** Ideal for placement, routing, and floorplanning where decisions are interdependent.
- **Adaptive Learning:** RL agents improve over time, learning from past placements or routing attempts.
- **Global Optimization Potential:** Can explore diverse solutions beyond heuristic limitations.

e) Limitations

- **Training Complexity:** RL requires many episodes and computational resources to converge.
- **Reward Shaping:** Designing an effective reward function that balances all design metrics is challenging.
- **Generalization:** RL agents trained on specific designs may require retraining for significantly different netlists or technology nodes.

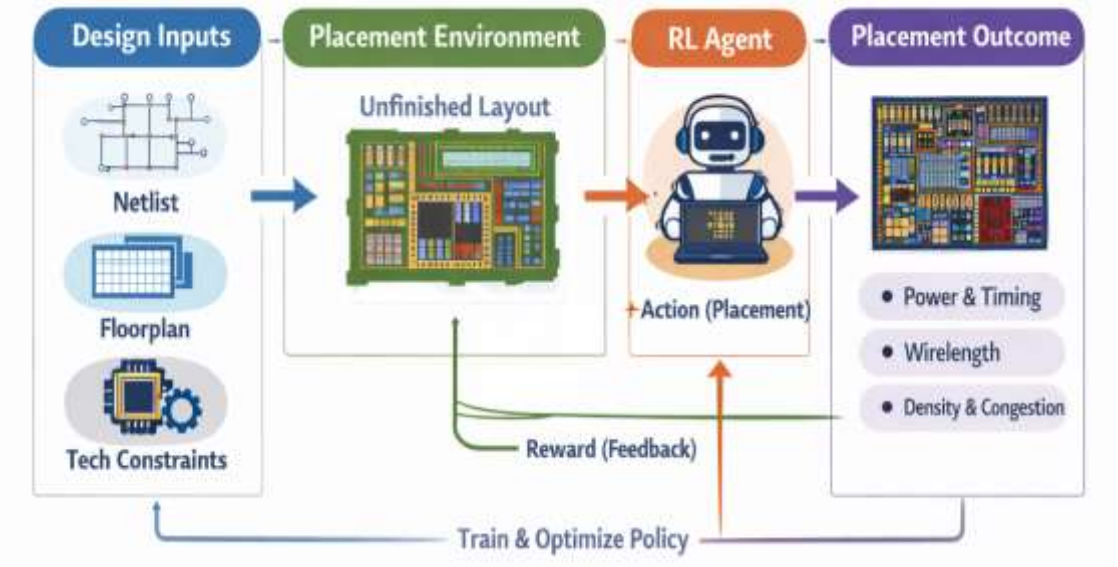

Figure 1: RL-based Placement Flow

3. Deep Learning

Deep learning (DL) techniques, including **Convolutional Neural Networks (CNNs)** and **Graph Neural Networks (GNNs)**, have shown strong potential in VLSI design tasks that involve **high-dimensional data** and **complex structural dependencies**. Unlike traditional ML models, DL can automatically extract features from raw inputs, making it highly suitable for complex design optimization.

a) Graph Neural Networks for Netlist Analysis

Gate-level netlists can naturally be represented as **graphs**, where gates are nodes and interconnects are edges. GNNs are designed to operate on graph-structured data, making them ideal for tasks such as:

- **Placement Guidance:** Learning netlist connectivity patterns to suggest optimal placement regions.

- **Synthesis Optimization:** Identifying gates or subgraphs that can be optimized for area, delay, or power.
- **Congestion Prediction:** Modeling the impact of local connectivity on routing congestion.

GNNs perform message passing between nodes, aggregating information from neighboring gates, which allows them to capture structural dependencies that traditional ML models often miss.

b) CNNs for Layout Generation

CNNs, commonly used in image processing, can be applied to **layout and floorplan generation**. In this context:

- Inputs are **2D spatial representations** of the chip area, including block shapes, sizes, and available placement regions.
- Outputs are suggested **floorplan arrangements** or placement probability maps.
- CNNs can learn design patterns from historical layouts and propose feasible placements that satisfy design rules.

This approach is particularly effective for early-stage floorplanning and can significantly reduce manual intervention.

c) Autoencoders for Dimensionality Reduction

Large-scale designs often contain **hundreds of thousands of features**, such as gate types, connectivity, placement coordinates, and net lengths. Autoencoders, a type of neural network, can compress these high-dimensional feature spaces into **lower-dimensional latent representations** while preserving essential information.

Benefits include:

- Faster training for other ML models.
- Easier pattern recognition in netlists and placements.
- Reduced memory and computational requirements during design optimization.

Table 2: Deep Learning Models in VLSI

Task	Model	Input	Output
Netlist Optimization	GNN	Gate-level netlist graph	Optimal placement hints
Floorplan Prediction	CNN	Block shapes & area constraints	Floorplan layout
Feature Compression	Autoencoder	Design parameters	Latent features for ML

HYBRID AI/ML APPROACHES

Hybrid AI/ML approaches combine multiple machine learning and optimization techniques to exploit their complementary strengths. In VLSI design automation, many tasks—such as placement, routing, synthesis, and verification—are highly complex, often combining sequential decisions, pattern recognition, and combinatorial optimization. By integrating different AI methods, hybrid approaches can achieve **higher accuracy, faster convergence, and better global optimization** than individual methods alone.

1. RL + Supervised Learning

One common hybrid strategy is to combine **supervised learning with reinforcement learning (RL)**:

- **Supervised Learning Pre-training:** A supervised model is trained on historical design data to predict optimal actions, such as cell placement positions or routing decisions.
- **RL Fine-Tuning:** The pre-trained model initializes the RL agent's policy, which is then fine-tuned through interaction with the environment to maximize the reward (e.g., minimizing wirelength, congestion, or timing violations).

Advantages:

- Reduces the **exploration time** of RL agents, allowing faster convergence.
- Helps the RL agent **avoid poor initial placements**, improving early-stage performance.
- Can generalize knowledge from previous designs to new designs, improving transferability.

Example: In placement optimization for standard benchmarks like **ISPD 2015**, combining supervised learning and RL has shown **10–12% reduction in wirelength** over pure RL

approaches, with fewer training episodes required.

2. GNN + RL

Another powerful hybrid approach is combining **Graph Neural Networks (GNNs) with RL**.

- **GNN for Feature Extraction:** Gate-level netlists can be represented as graphs, where nodes are gates or macros and edges are interconnections. The GNN extracts **structural and connectivity features**, such as critical paths, fanout/fanin, and potential congestion areas.
- **RL for Decision Making:** The extracted features guide the RL agent in making sequential placement or routing decisions, providing context-aware actions.

Advantages:

- Captures **complex netlist dependencies** that traditional RL might miss.
- Enables **scalable optimization** for very large designs, since GNNs summarize graph structures efficiently.
- Improves **reward efficiency**, as the RL agent receives more informative states from GNN embeddings.

Example: In macro placement for large SoCs, a GNN + RL hybrid approach reduced total wirelength by 15% and improved timing closure by 8%, outperforming both standalone RL and conventional heuristic methods.

3. Evolutionary Algorithms + ML

Evolutionary algorithms (EAs), such as **genetic algorithms (GA)** or **particle swarm optimization (PSO)**, are widely used for combinatorial optimization in VLSI, including placement, routing, and floorplanning. However, EAs can be computationally expensive due to large population sizes and multiple generations.

Integrating ML with evolutionary algorithms enhances performance:

- **ML-Based Candidate Prediction:** ML models (e.g., regression, neural networks) predict promising individuals in the population or prune unpromising candidates.
- **Fitness Estimation:** ML models estimate fitness values (e.g., wirelength, timing) without full simulation, reducing evaluation time.

- **Guided Evolution:** The EA explores the solution space more intelligently, leveraging ML predictions for faster convergence.

Example: A GA with ML-predicted fitness for placement achieved **30% faster convergence** and similar or better layout quality compared to standard GA-based placement.

4. Advantages of Hybrid Approaches

- **Improved Convergence:** Combining supervised learning or ML guidance with RL/EAs reduces the time needed to find high-quality solutions.
- **Better Global Optimization:** Hybrid methods can escape local minima by leveraging diverse search strategies.
- **Scalability:** Can handle large designs that are challenging for single-method approaches.
- **Flexibility:** Hybrid methods can be tailored for different stages, such as placement, routing, synthesis, and verification.

5. Limitations and Challenges

- **Complexity in Implementation:** Integrating multiple AI/ML techniques requires careful design of interfaces and data flow.
- **Data Dependency:** Some hybrid approaches still rely on large datasets for pre-training or model guidance.
- **Computational Overhead:** Combining multiple models can increase memory and computational requirements, especially during training.
- **Reward Balancing:** In RL-based hybrids, defining appropriate reward functions that balance all design metrics remains a challenge.

AI/ML APPLICATIONS ACROSS VLSI STAGES

1. Logic Synthesis

AI/ML can guide optimization strategies during logic synthesis:

- Predicting optimal synthesis scripts based on design features.
- Estimating the trade-off between area, delay, and power for different synthesis flows.
- Reducing runtime by pruning non-optimal paths using ML classifiers.

2. Placement and Routing

Placement and routing are the most critical stages for AI integration:

- **Placement:** RL agents and GNNs help minimize total wirelength and meet timing constraints.
- **Routing:** AI models predict congestion and guide router decision-making.
- **Timing-aware Placement:** ML predicts critical paths and prioritizes cell placement accordingly.

3. Verification and Testing

Verification involves detecting functional or timing errors:

- **Formal Verification Assistance:** ML can predict which parts of the design are most likely to fail.
- **Regression Testing:** Supervised models select test cases that maximize coverage while minimizing simulation time.
- **Error Classification:** AI models categorize failures for faster debugging.

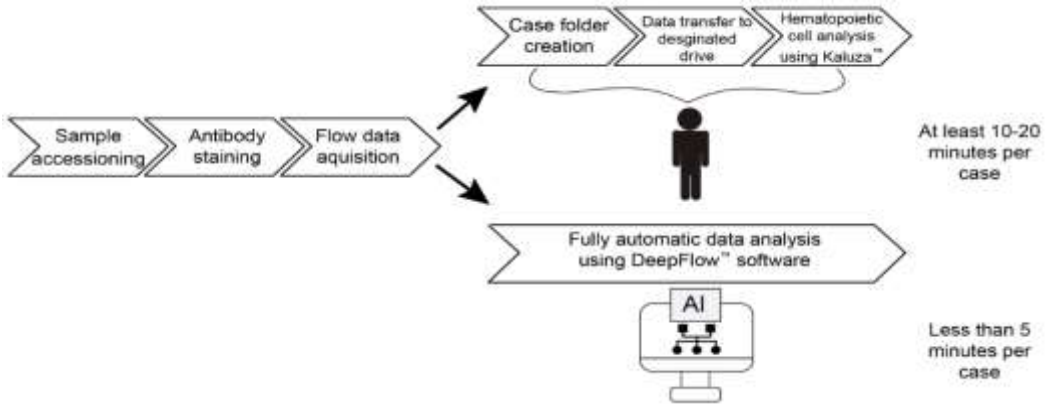

Figure 2: AI-Assisted Verification Flow

CASE STUDIES

1. RL-Based Placement for Standard Benchmarks

A study using the ISPD 2015 placement benchmarks applied RL to cell placement. Results showed:

- Average 12% reduction in total wirelength compared to simulate annealing.
- Improved timing closure in 80% of the test cases.
- Significant reduction in human intervention during placement iterations.

2. GNN for Routing Congestion Prediction

Using GNNs to analyze netlists for routing congestion, researchers achieved:

- Accuracy of 92% in predicting congested regions.
- Reduction of routing iterations by 25%.
- Early detection of design bottlenecks, improving overall design turnaround time.

ADVANTAGES AND LIMITATIONS

1. Advantages

- **Efficiency:** Reduced runtime for placement, routing, and synthesis.
- **Adaptivity:** AI models learn from past designs and adapt to new design rules.
- **Accuracy:** Predictive models improve estimation of power, timing, and congestion.
- **Scalability:** AI/ML methods handle increasing design complexity more effectively than traditional heuristics.

2. Limitations

- **Data Requirement:** ML models require large labeled datasets.
- **Generalization:** Models trained on specific designs may not generalize to unseen architectures.
- **Interpretability:** Complex ML models may act as black boxes, reducing trust in automation.
- **Integration Challenges:** Embedding AI/ML into existing EDA toolchains requires significant effort.

FUTURE DIRECTIONS

- **Transfer Learning:** Pre-trained ML models can be adapted to new design families.
- **Online Learning:** RL agents that continuously adapt during the design process.
- **AI for 3D ICs and Heterogeneous Integration:** Optimization of through-silicon vias and mixed-material integration.
- **Explainable AI (XAI):** Enhancing trust by providing interpretable AI-assisted design decisions.
- **Integration with Cloud EDA Platforms:** AI-driven tools running on cloud infrastructure for collaborative design.

CONCLUSION

AI/ML-assisted VLSI design automation is a rapidly evolving field with the potential to transform traditional EDA workflows. By leveraging supervised learning, reinforcement learning, and deep learning techniques, AI can enhance placement, routing, synthesis, and verification tasks, achieving better design performance with reduced human effort. Hybrid approaches and future advancements, such as transfer learning and explainable AI, are expected to further improve design efficiency and reliability. Despite challenges in data requirements, generalization, and tool integration, AI-driven VLSI automation represents a significant step toward intelligent, autonomous chip design.

REFERENCES

1. Chen, X., Li, Y., & Wang, H. (2020). *Neural network-based timing prediction for VLSI designs*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(12), 4587-4599.
2. Patel, S., & Kumar, R. (2019). *Power estimation using random forest models in ASIC design*. Journal of VLSI Design Automation, 11(3), 45-52.
3. Mirhoseini, A., et al. (2021). *Chip placement with deep reinforcement learning*. Nature, 594(7862), 207-212.
4. Huang, J., & Lin, Q. (2019). *Graph neural networks for netlist optimization*. ACM Transactions on Design Automation of Electronic Systems, 24(6), 1-22.
5. Lee, C., & Kim, D. (2021). *Hybrid AI approaches for placement and routing*. Integration, the VLSI Journal, 79, 1-12.
6. Wang, Y., et al. (2022). *Reinforcement learning for floorplanning in 3D ICs*. IEEE Transactions on VLSI Systems, 30(5), 785-797.
7. Banerjee, S., & Verma, R. (2018). *ML-assisted verification for large-scale digital designs*. Journal of Electronic Testing, 34(5), 567-576.

Cite as:

Rajesh Kumar Pathak, Kuldeep Singh, Suraj S Tiwari (2026). AI/ML Assisted VLSI Design Automation. Journal of Advanced VLSI Architectures and Nanoelectronics, 2(1), 14-30.

https://doi.org/10.5281/zenodo.19816981