Politecnico Di Torino

Master's Degree in Electronic Engineering

Master's Degree Thesis

Survey on Machine Learning and Artificial Intelligence used for Electronic Design Automation

Supervisor:

Prof. Luciano.Lavagno

Candidate:

Xing Zeyuan (296454)

July 2024

Contents

CHAPTER 1

Abstract

The complexity of EDA tools for ICs are crucial enablers for the semiconductor industry as the size of integrated circuits (ICs) have been increasing enormously. With groundbreaking innovations in IC design and integration, some chips even have up to billions transistors, in the meanwhile the slowing down of Moore's Law have caused that the number of transistors per design increases exponentially and doubles every two years. Consequently, the corresponding design space which originally supposed to be searched for an implementation that satisfies all specifications and then optimizes all related factors concerned as NP-Hardness problem in EDA like power, area, delay (PPA) and runtime, etc. Along side this phenomenon, Machine Learning (ML) based algorithms which could be used to enhance EDA tools and processes as Functional simulation, Logical synthesis, Physical design (Placement & Routing mainly included) and some specific techniques that used to do verification and test as Formal verification. Machine Learning techniques have been employed in many domains with great success because of their ability to build powerful models from data. Consequently, ML has also been applied in computer engineering where it guarantees to complement the insufficiency left by heuristic algorithms and start other new possibilities. Employing ML-based algorithms gives the designers space to cultivate the abstraction level by concentrating to the objective itself only and leave the practical details on how to reach the ultimate goals to the ML models.

This thesis provides a comprehensive survey of the specific steps and techniques mainly used in state-of-art EDA tools that use machine learning algorithms. The ML-based EDA tools are categorized based on the IC design steps and techniques which are functional simulation, formal verification, logic synthesis, placement and routing separately. State-of-the-art ML-based VLSI-EDA tools, current trends, and future perspectives of ML in VLSI-EDA are also discussed in the end.

CHAPTER 2

Introduction

Over time, the Integrated Circuits (ICs) chip design flow has merged with several software tools to logic synthesis, functional simulation, physical design, test and verify different electronic designs efficiently.The overall compendium of all those technologies is called Electronic Design Automation (EDA). While the down-scaling of CMOS technology, the design complexity of very large-scale integrated is incrementing, which leads to the insufficiency of traditional EDA tools, Electronic Design Automation is a critical field in the semiconductor industry, responsible for the automation of IC design processes.

As IC designs become more intricate and multifaceted, traditional EDA tools encounter significant challenges in terms of efficiency, accuracy, and scalability. This is particularly evident with the ongoing advancement of Moore's Law, which has resulted in ICs with exponentially growing scale and complexity. The traditional human-driven design and verification processes are becoming increasingly time-consuming and less precise, creating a pressing need for Artificial Intelligence (AI) intervention to handle tasks that are becoming infeasible for manual completion due to their time-consuming nature and inherent low precision. Furthermore, the current EDA approaches are time and resource consuming with no optimal-solution guaranteed usually. However, the design has to be verified and tested to ensure correctness, or to do the prediction before the actual synthesis to achieve optimal PPA (Power, Performance, Area) and save the multiple iteration times in the meanwhile. To achieve better performance and alleviate the situation of EDA tools, AI and Machine Learning (ML) has been incorporated into many stages of design flow.

The rapid advancements in Machine Learning (ML) and Artificial Intelligence (AI) have revolutionized numerous domains, including Electronic Design Automation (EDA). EDA tools, essential for designing complex integrated circuits (ICs), have traditionally relied on algorithmic techniques. However, the ever-increasing complexity of IC designs necessitates more efficient and intelligent solutions. This has led to the integration of ML and AI techniques across various stages of the EDA process, from functional simulation to routing.

Machine Learning (ML) and Artificial Intelligence (AI) offer promising solutions to these challenges by providing advanced methods for optimization, prediction, and automation. These technologies can significantly enhance the efficiency and accuracy of EDA tools, making them indispensable for future IC design processes. As the scale and complexity of integrated circuits continue to grow, the necessity for innovative approaches to design automation becomes even more crucial. The integration of AI and ML into EDA processes aims to address these challenges by automating the traditionally labor-intensive and error-prone tasks, thus improving overall design productivity and accuracy.

Functional simulation is one of the initial stages in the EDA process where ML techniques have been applied to improve efficiency. Traditional simulation methods often require exhaustive testing, which can be both time-consuming and resource-intensive. ML techniques, such as isolation-forest anomaly detection, have been employed to select the most relevant tests for simulation, thereby reducing the overall runtime without compromising coverage. This approach not only enhances the efficiency of the simulation process but also ensures that critical functional behaviors are thoroughly tested. By leveraging ML algorithms, engineers can focus on the most pertinent test cases, thereby accelerating the verification process and reducing the computational burden.

In addition to enhancing test selection, ML techniques have been used to predict potential simulation outcomes, enabling more targeted testing strategies. For instance, predictive models can identify likely failure points within a design, allowing engineers to preemptively address issues before they become critical. This proactive approach not only saves time but also enhances the reliability and robustness of the final product. The ability to predict and mitigate potential issues early in the design cycle is a significant advantage, particularly as IC designs become more complex and prone to subtle, hard-to-detect faults.

Formal verification is another crucial stage in the EDA process where ML techniques have shown significant promise. Boolean Satisfiability (SAT) solvers, which are fundamental to formal verification, have been augmented with various ML techniques to improve their performance. Techniques such as Gated Graph Convolutional Networks (Gated-GCN), Recurrent Graph Neural Networks (Recurrent-GNN), and Reinforcement Learning (RL) have been utilized to develop more efficient SAT solvers. These ML-enhanced solvers can predict solutions, guide local search heuristics, and learn branching heuristics, thereby making the formal verification process more robust and scalable. The integration of ML in formal verification not only accelerates the process but also enhances its accuracy, enabling the verification of increasingly complex designs within feasible timeframes.

One of the significant challenges in formal verification is the state space explosion problem, where the number of possible states that need to be checked grows exponentially with the size of the design. ML techniques, particularly those involving neural networks and reinforcement learning, offer a way to navigate this vast state space more efficiently. By learning from previous verification tasks, these techniques can identify patterns and heuristics that lead to quicker convergence on a solution. This capability is especially valuable for verifying modern, large-scale IC designs, where traditional methods struggle to keep pace with the complexity.

Logic synthesis, a critical phase in the EDA workflow, has also benefited from the integration of ML techniques. Power estimation, Quality of Results (QoR) improvement, and design space exploration are key aspects of logic synthesis where ML has been applied. Graph Neural Networks (GNN), Deep Neural Networks (DNN), and Recurrent Neural Networks (RNN) are among the ML techniques used to develop models that can accurately estimate power consumption, improve QoR, and automate the design space exploration process. These advancements have led to more efficient and effective logic synthesis methodologies. ML-driven logic synthesis enables designers to explore a wider array of design alternatives more quickly, leading to optimal solutions that might be missed using traditional methods.

Moreover, ML techniques have been employed to optimize the trade-offs between power, performance, and area (PPA) in logic synthesis. By training models on large datasets of design examples, these techniques can predict the PPA characteristics of new designs with high accuracy, enabling more informed decision-making. This predictive capability is particularly valuable in the early stages of design, where rapid iterations are essential to meet stringent design constraints. The ability to accurately estimate and optimize PPA characteristics early in the design process significantly enhances the overall efficiency and effectiveness of the EDA workflow.

Placement and routing are subsequent stages in the EDA process that involve the physical arrangement of circuit components on a chip. The optimization of placement and routing is critical for minimizing wire length, improving timing, and ensuring overall chip performance. ML techniques such as Graph Attention Networks (GAT), Policy Gradient Optimization, and Convolutional Neural Networks (CNN) have been employed to optimize these stages. These techniques enable more accurate predictions of wire length, timing, and congestion, thus facilitating better placement and routing decisions. The application of ML in placement and routing not only improves the quality of the final layout but also reduces the time and effort required to achieve an optimal design.

In the placement stage, ML techniques help in predicting the optimal placement of components to minimize wire length and improve overall design performance. For instance, GNNs can model the interdependencies between various components and predict their optimal positions on the chip. This predictive capability reduces the need for iterative placement adjustments, thereby speeding up the design process. Additionally, ML techniques can be used to optimize specific design parameters, such as timing and power consumption, by considering a broader range of placement options than traditional methods.

Routing, the final stage in the EDA process, is where the physical connections between components are established. ML techniques have been particularly effective in congestion estimation and routability prediction. Linear Regression, Artificial Neural Networks (ANN), Random Forests, and more advanced models like Conditional Generative Adversarial Networks (Conditional-GAN) and Lattice Hypergraph Neural Networks (LHNN) have been used to predict routing congestion and improve the routability of designs. These techniques help identify potential routing issues early in the design process, allowing for preemptive adjustments that enhance overall design quality and performance. By accurately predicting and mitigating routing challenges, ML techniques ensure that the final design meets all performance and reliability requirements.

Furthermore, ML techniques have been used to develop routing algorithms that adapt to the specific characteristics of each design. For example, reinforcement learning-based approaches can learn from previous routing tasks to develop strategies that are tailored to the unique requirements of new designs. This adaptability is crucial for handling the diverse and increasingly complex routing challenges posed by modern ICs. By continuously learning and improving, these ML-based routing algorithms provide a dynamic and effective solution to one of the most challenging aspects of EDA.

The integration of ML and AI in EDA represents a significant paradigm shift, promising to enhance the efficiency and effectiveness of IC design processes. As AI and ML technologies continue to evolve, they are expected to become increasingly indispensable in EDA design, leading to higher efficiency and greater precision in the design and verification of ICs. This survey aims to provide researchers and practitioners with insights into current trends, methodologies, and potential future developments in this rapidly evolving field. By summarizing the cutting-edge research that combines AI and ML with EDA design, this survey seeks to highlight the transformative potential of these technologies in overcoming the challenges posed by modern IC design complexities.

Through this comprehensive review, we aim to shed light on the significant advancements and applications of ML and AI in EDA, focusing on their impact across various stages of the design process. By examining the state-of-the-art research and identifying key trends and future directions, this survey serves as a valuable resource for both academic researchers and industry practitioners seeking to leverage ML and AI for more efficient and effective EDA solutions. The ultimate goal is to provide a thorough understanding of how these technologies are currently being applied, their benefits, and their limitations, as well as to offer insights into future research directions that can further enhance the capabilities and applications of ML and AI in EDA.

The transformative potential of ML and AI in EDA is particularly evident when considering the broader implications for the semiconductor industry. As the demand for more powerful and efficient ICs continues to grow, the pressure on design teams to deliver innovative solutions within shorter time-slots is increasing. ML and AI offer a path to meet these demands by automating and optimizing key aspects of the design process. This not only reduces the time and resources required to bring new products to market but also enables the development of more advanced and reliable ICs.

The integration of ML and AI into EDA processes is not just about replacing existing techniques but augmenting them with new capabilities. For instance, ML models can be used to analyze vast amounts of design data, uncovering patterns and insights that human designers might miss. This can lead to more informed decision-making and better design outcomes. Additionally, AI-driven tools can continuously learn and improve from new data, ensuring that they remain effective even as design challenges evolve. This adaptability is crucial in a field where technological advancements occur at a rapid pace, requiring tools that can keep up with the latest developments and provide cutting-edge solutions.

Moreover, the use of ML and AI in EDA can facilitate greater collaboration and integration across different stages of the design process. For example, data from the functional simulation stage can be used to inform formal verification, logic synthesis, placement, and routing, creating a more cohesive and streamlined workflow. This holistic approach can help identify and address potential issues earlier in the design process, reducing the likelihood of costly revisions later on. By breaking down isolations between different EDA stages, ML and AI can enable a more efficient and effective design process that leverages the full potential of these advanced technologies.

In conclusion, the integration of ML and AI into EDA processes is a crucial development that addresses the growing complexity and demands of modern IC design. By leveraging these advanced technologies, the EDA field can achieve significant improvements in efficiency, accuracy, and scalability, paving the way for the next generation of semiconductor innovations. This survey aims to provide a comprehensive overview of the current state of ML and AI applications in EDA, highlighting the key research advancements, practical applications, and future directions. By doing so, it seeks to contribute to the ongoing evolution of EDA tools and methodologies, ultimately supporting the continued progress of the semiconductor industry. The ongoing research and development in this field promise to unlock new possibilities for IC design, enabling the creation of more powerful, efficient, and reliable electronic devices that meet the ever-increasing demands of modern technology.

Through this detailed exploration, we hope to inspire further research and innovation in the application of ML and AI to EDA. As these technologies continue to advance, their integration into EDA processes will become increasingly sophisticated, offering even greater benefits. By staying at the forefront of these developments, the semiconductor industry can continue to push the boundaries of what is possible, driving progress and delivering cutting-edge solutions that shape the future of technology. The potential of ML and AI in EDA is vast, and we are only beginning to scratch the surface of what these technologies can achieve. With continued investment and collaboration, the future of IC design looks brighter than ever, promising a new era of innovation and excellence in the semiconductor industry.

The rest of the paper is organized as follows. In chapter 3 I described background information on different Neural Networks as Graph Neural Network, Convolutional Neural Network and Reninforcement Learning

Methods briefly. Chapter 4 provides a comprehensive illustration of EDA steps and techniques that utilized machine learning based methods to solve different tasks and reach specific purposes. At last in chapter 5 I gave a conclusion of the whole paper and present a personal perspective of the future concerns the ML-based methods within EDA that drew from those papers.

CHAPTER 3

Background Information

In this chapter, I briefly make a summary of background information related to Reinforcement Learning, Graph Neural Network and Convolutional Neural Network.

3.1 Graph Neural Network (GNN)

In the context of advanced computational techniques, Graph Neural Networks (GNNs) have emerged as a powerful tool for handling graph-structured data, which is prevalent in many domains, including EDA. GNNs extend traditional neural networks by incorporating graph structures into their computations, allowing them to capture dependencies and relationships between nodes. A graph G = (V,E) consists of a set of nodes V and edges E that connect these nodes. In the case of EDA, the nodes could represent components such as logic gates, while the edges represent the connections between them.

GNNs operate by iteratively updating the representation of each node based on the features of its neighboring nodes. This process can be mathematically described by the following update rule:

$$h_v^{(k)} = \sigma(\sum_{\mu \in N_{(v)}} W^{(k)} h_\mu^{(k-1)} + b^{(k)}) \tag{3.1}$$

Where

- $h_v^{(k)}$ represents the feature vector of node v at iteration k;

- $N_{(v)}$ denotes the set of neighboring nodes of v;
- $W^{(k)}$ is a learnable weight matrix;
- $b^{(k)}$ is a bias term;
- σ is an activation function;

This equation captures the essence of message passing in GNNs, where information from neighboring nodes is aggregated to update the node's representation.

The power of GNNs lies in their ability to learn complex dependencies and relationships within graph-structured data. This makes them particularly well-suited for applications in EDA, where the design and verification processes often involve data that can be naturally represented as graphs. For instance, in circuit design, the components and their interconnections form a graph structure that can be effectively analyzed using GNNs. By leveraging the structural information in the graph, GNNs can perform tasks such as predicting design characteristics, optimizing placement and routing, and improving formal verification.

Another key advantage of GNNs is their ability to generalize across different graph structures. This means that a GNN trained on a specific set of graphs can be applied to new, unseen graphs, making them highly versatile and adaptable. This generalization capability is particularly valuable in EDA, where designs can vary significantly in complexity and structure. GNNs can learn to capture the underlying principles and patterns that govern the behavior of different designs, enabling them to provide insights and optimizations that are broadly applicable.

3.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are another class of neural networks widely used for processing grid-like data, such as images. CNNs are particularly effective at capturing spatial hierarchies and patterns through the use of convolutional layers. A convolutional layer applies a set of filters to the input data, generating feature maps that highlight various aspects of the data. The convolution operation for a single filter can be expressed as:

$$(f * x)(i, j) = \sum_m \sum_n x(i + m, j + n)\omega(m, n) \tag{3.2}$$

Where

- x is the input image;
- ω is the fileter;
- (i,j) are the coordinates of the output feature map;

The filters are learned during the training process, allowing the network to detect features such as edges, textures, and shapes at different levels of abstraction.

CNNs also employ pooling layers, which reduce the spatial dimensions of the feature maps and help to make the representations more invariant to small translations of the input. The most common type of pooling is max pooling, which takes the maximum value in each local region of the feature map:

$$y(i,j) = max_{(m,n)\in R(i,j)} x(m,n) \tag{3.3}$$

Where

- R(i,j) denotes the local region around the coordinate(i,j);

The combination of convolutional and pooling layers allows CNNs to build increasingly abstract and high-level representations of the input data, which are then used for various tasks such as classification, object detection, and segmentation. In the context of EDA, CNNs can be applied to tasks where the design data can be represented as images or matrices, such as layout optimization, thermal analysis, and hotspot detection.

CNNs have shown remarkable success in a wide range of image processing tasks, and their application in EDA is an extension of this success. For example, in layout optimization, CNNs can analyze the layout of the circuit to identify areas that need improvement and suggest changes that can enhance the overall performance of the design. Similarly, in thermal analysis, CNNs can predict the thermal behavior of the circuit based on its layout and component placement, allowing designers to make adjustments that reduce hotspots and improve thermal management.

In hotspot detection, CNNs can identify regions of the design that are likely to cause manufacturing issues, such as areas with high density of vias or narrow wire spacing. By detecting these hotspots early in the design process, designers can make necessary adjustments to ensure that the design meets manufacturing constraints and reduces the likelihood of defects. This capability is particularly important in advanced technology nodes, where the margin for error is significantly smaller, and even minor issues can lead to yield loss.

3.3 Reinforcement Learning (RL)

Reinforcement learning (RL) is a type of machine learning that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences. Unlike supervised learning where a training dataset complete with correct inputs and outputs is provided, in reinforcement learning, the agent is designed to learn from the consequences of its actions. This approach is closely aligned with the ways in which sentient beings learn from the environment, making it particularly useful for applications where explicit programming is not feasible.

The foundational concept of RL involves understanding what actions an agent should take in a given environment to maximize a notion of cumulative reward. The agent, through interactions within a designed or simulated environment, receives rewards by performing correctly and penalties for making errors, guiding it to learn optimal behaviors inherently. This method is modeled as a Markov Decision Process (MDP), characterized by states, actions, rewards, and transitions.

MDP

- States(S) : A set of states that the enviroment can be in.
- Actions(A) : A set of actions that the agent can take.
- Transition Probability(P) : P(s'|,s,a) defines the probability of transitioning from state s to state s' after taking action a.
- Reward Function(R) : R(s,a,s') defines the immediate reward received after transitioning from state s to state s', due to action a.
- Discount Factor(γ) : A factor γ that between 0 and 1 and discounts future rewards, reflecting the preference for sooner rather than later rewards.

An RL model consists of three primary components: a policy, a reward signal, and a value function. The policy, often denoted as π, defines the behavior of the agent by mapping states of the environment to the actions to be taken when in those states. The reward signal, pivotal to reinforcement learning, indicates the immediate return from an action taken by the agent. The goal of the agent is often to maximize the total reward it receives over the long run. The value function specifies what is good in the long run; it provides a prediction of future rewards that can be expected, conditioned on the current state and action taken by the agent.

Training an RL agent involves exploration and exploitation, where the agent has to balance between exploring new actions that might lead to higher rewards and exploiting the actions that are known to yield the best results. This dilemma is often handled using strategies such as ϵ-greedy, where the agent will usually take the best known action but occasionally will explore random actions. Another method is the use of advanced algorithms like Upper Confidence Bound (UCB) or Thompson sampling which intelligently balance this trade-off based on uncertainty and reward potential.

The learning process updates the policy based on the observed rewards and penalties. Algorithms such as Q-learning and policy gradient methods are utilized to achieve this. In Q-learning, the agent learns an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the current policy thereafter. For each state-action pair, it maintains a Q-value which is updated using the rewards obtained. In policy gradient methods, instead of learning a value function that tells the potential of each action, the parameters of the policy itself are adjusted in the direction that maximizes the expected reward.

Deep reinforcement learning combines neural networks with a reinforcement learning architecture that enables agents to make decisions from unstructured input data. By using deep learning, the agent can interpret complex sensors and image data to make informed decisions. This has led to significant breakthroughs, particularly in areas like gaming, autonomous driving, and robotics, where traditional approaches were limited.

Recent advancements in RL include the integration of deep learning techniques, leading to the development of Deep Q-Networks (DQN) that have achieved superhuman performance in games like Atari. Beyond games, reinforcement learning is applied in various real-world scenarios such as robotic control, energy management, healthcare, and more, demonstrating its broad applicability and potential.

Despite its potential, RL faces challenges such as high variance in outcomes, substantial data requirements for training in complex scenarios, and the difficulty in defining appropriate rewards that effectively guide the agent towards desired behaviors without unintended side effects. Moreover, RL systems are typically computationally expensive and sensitive to hyperparameters, which necessitates careful tuning and substantial computational resources.

In summary, EDA and advanced neural network architectures like GNNs and CNNs represent two fundamental areas of modern semiconductor design. EDA provides the tools and methodologies to automate the design and verification of ICs, ensuring that they meet the required specifications and performance criteria. GNN, RL and CNNs, on the other hand, offer powerful techniques for processing and analyzing complex data structures, enabling new levels of efficiency and accuracy in various tasks. By understanding the foundational concepts of both EDA and these neural networks, researchers and practitioners can better leverage their capabilities to address the growing challenges of modern IC design.

CHAPTER 4

ML for EDA

4.1 ML for Functional Simulation

Functional simulation in Electronic Design Automation (EDA) serves as a foundational stage, ensuring integrated circuits (IC) designs meet specified logical correctness before entering the more intricate phases of development. This simulation validates the IC's behavior under varied operational scenarios to catch any functional inaccuracies early, thereby saving significant time and resources later in the design cycle.

The employment of machine learning (ML) techniques in functional simulation has introduced a revolutionary approach to handling the vast data and complex scenarios typical of this stage. Particularly, the use of unsupervised learning models like isolation forests in Reference [1] to detect anomalies in test coverage data exemplifies a strategic enhancement over traditional methods. These models analyze the outputs from fast functional simulators, which provide rapid feedback on the IC's behavior under test conditions. By focusing on tests as shows in Figure 4.1that uncover unique or infrequent behaviors, ML-driven approaches can drastically reduce the number of tests needed while still maintaining thorough coverage. This method not only cuts down on the computational load but also speeds up the overall simulation process by prioritizing areas of the design that are most likely to contain critical faults. However, while ML methods offer significant efficiency improvements, they do require careful calibration and a thorough understanding of the underlying data to be effective. The quality of the results depends heavily on the accuracy of the anomaly detection model and the relevance of the features it considers.

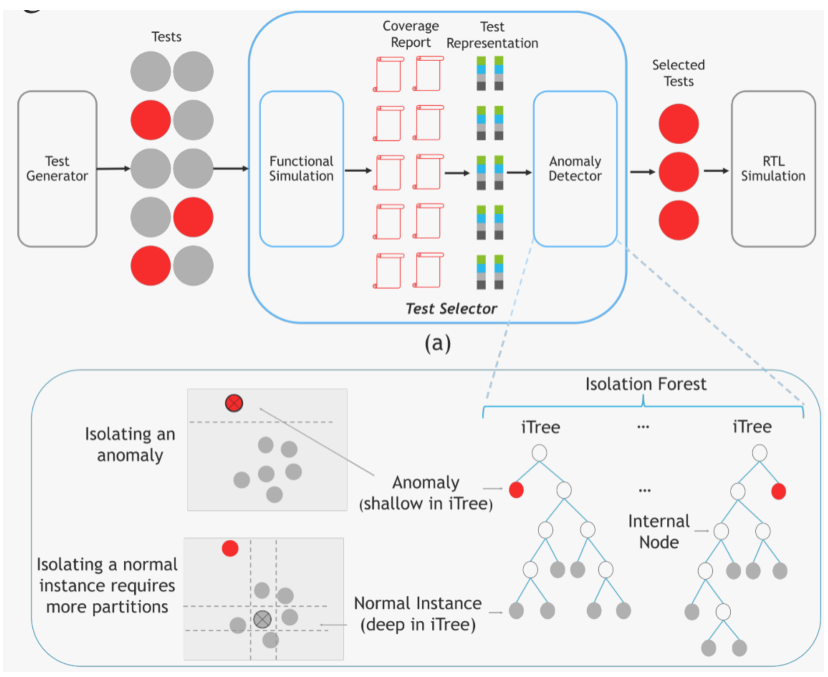

Figure 4.1: Test Selection Flow with isolation-forests machine learning method as an Anomaly Detector

There is also the risk of overlooking important issues if the model is not appropriately tuned to recognize subtle but critical anomalies.

Nevertheless, due to the characteristic of functional simulation at RTL which is "100%-accurate", the novel techniques like Machine Learning is nearly impossible to optimize this part directly with guarantee the accuracy baseline, hence I looked through several papers that dedicate in this purpose with traditional methods.

Traditional methods in functional simulation continue to play a crucial role, particularly through advancements in optimizing compilers and code instrumentation techniques. For example, optimizing compilers like the one used in Reference [2] the JIT compilation approach for RTL simulation can significantly enhance simulation speeds. By applying hardware-centric optimizations and leveraging an advanced intermediate representation that accurately models the hardware's behavior, these compilers transform how simulation tasks are executed. They allow for runtime code generation that is highly optimized for the target architecture, resulting in faster simulation times.

Code instrumentation techniques, particularly those that focus on loop-oriented optimizations, address another common bottleneck in RTL simulations. In Reference [3]by identifying and optimizing the performance of critical loops within the simulation code, these techniques can considerably reduce the time required to simulate complex designs. They provide deep insights into the code's execution at runtime, enabling targeted optimizations that improve overall simulation efficiency.

Despite their benefits, traditional methods also have limitations. For instance, optimizing compilers require extensive knowledge of both software and hardware architectures to effectively translate RTL code into an optimized form. This dual requirement can complicate the development and maintenance of such tools. Similarly, code instrumentation can introduce overheads that may slow down the simulation under certain conditions, especially if not carefully managed.

Moreover, the integration of both ML techniques and traditional methods presents a comprehensive approach to tackling the challenges of functional simulation. ML methods provide a high-level overview, identifying key areas for attention, while traditional techniques offer low-level optimizations that enhance the execution of simulation tasks. This dual approach ensures not only extensive coverage through intelligent test selection but also efficient processing of simulations, accommodating the increasing complexity and scale of modern IC designs.

Stage of EDA	Purpose	ML method
Functional Simulation	Runtime Reduction (by selecting tests for simulation)	isolation-forest anomaly detection [1]

Table 4.1: Summary of ML method applied in Functional Simulation

Incorporating ML into functional simulation marks a significant shift towards more intelligent and adaptive EDA tools. However, it also necessitates ongoing research to further refine these techniques and fully integrate them into the EDA workflow. As IC designs continue to grow in complexity, the synergy between ML-driven strategies and traditional optimization methods will be crucial in developing the next generation of EDA tools that are both powerful and efficient enough to meet the industry's evolving needs. This blend of old and new methodologies enriches the toolbox available to engineers, promising continued improvements in the speed and accuracy of IC design and verification processes.

4.2 ML for Formal Verification

The integration of machine learning (ML) into formal verification within the realm of Electronic Design Automation (EDA) is showing enormous potential, offering new strategies to improve the verification process's depth and efficiency. As the complexity of integrated circuits increasing promptly, traditional verification methods are often strained by the vast state spaces and intricate designs they need to manage. ML offers a suite of techniques that are increasingly considered essential in navigating these challenges more effectively.

Within which, the tasks that should be solved usually would be the Boolean SATisfiability problem, Model checking, Assertion Generation and Run-time Estimation task.

4.2.1 SAT

To start with Boolean SAT problem, which is a significant NP-complete problem with numerous practical applications–product configuration, hardware verification, and software package management, to name a few. There is no single efficient algorithm that solves every SAT problem, but heuristics have been developed that are sufficient for solving many real-life tasks.

A novel approach, as proposed by Zhang et al. [4], involves the use of a neural network model, NLocalSAT, to enhance stochastic local search (SLS) solvers for Boolean satisfiability problems, a cornerstone in formal verification.

NLocalSAT introduces a groundbreaking integration where a neural network is employed to predict initialization assignments for the SLS solvers, fundamentally altering the search dynamics from traditional random initialization methods. This method leverages the ability of neural networks to process complex patterns and predict outcomes, thereby providing a more targeted and efficient starting point for the SAT solvers. The approach significantly improves solver performance, achieving a 27% to 62% improvement in problem-solving efficiency over traditional methods.

The core of NLocalSAT involves a graph-based neural network model that converts SAT instances into a bipartite graph format, facilitating the neural network to better understand and process the structural information of the problem. This structure is then used to predict initial variable assignments that are more likely to lead to a solution rapidly. The integration of such predictive modeling into the initialization phase of SLS solvers addresses one of the major inefficiencies in SAT solving — the randomness of initial conditions, which often leads to prolonged search times and increased computational overhead.

Furthermore, the effectiveness of NLocalSAT is underscored by its application to various SLS solvers across

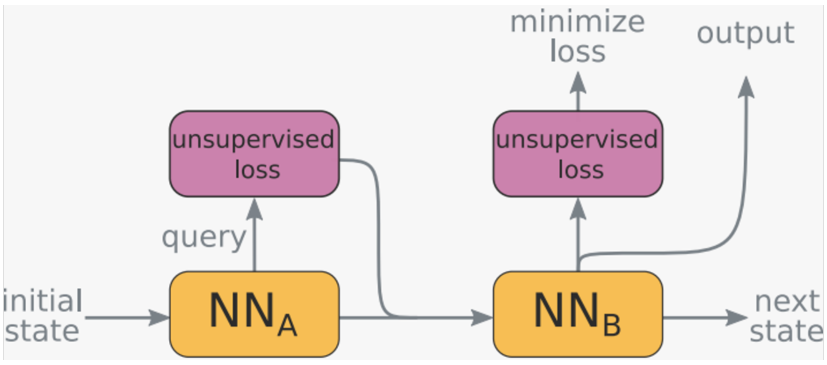

Figure 4.2: Query Machanism with an unsupervised Loss-function utilized

multiple datasets, showing consistent improvements across different types of SAT problems, which are common in formal verification tasks. This adaptability suggests that the NLocalSAT approach could be broadly applicable in the optimization of other formal verification processes within electronic design automation, potentially leading to more efficient design cycles and faster time-to-market for complex electronic systems.

As in Reference [5] presented by Ozolins et al. exemplifies this progression by introducing QuerySAT, a neural SAT solver enhanced with a unique query mechanism as shows in Figure 4.2. This mechanism allows the solver to iteratively refine its predictions based on feedback from an unsupervised loss function, setting a new standard for adaptive, intelligent computational tools in EDA. QuerySAT differentiates itself by utilizing a step-wise recurrent structure where each iteration involves generating a query of variable assignments, evaluating these against a specially formulated unsupervised loss function, and updating the solver's state based on the feedback. This feedback loop enables the model to continuously learn and adjust its strategies, mimicking a form of 'intelligent' problem-solving that evolves as the solver processes more information about the SAT instance it tackles.

The significant advantage of QuerySAT lies in its ability to integrate seamlessly into existing SAT solving frameworks while providing substantial improvements in performance. In comparative studies highlighted in the paper, QuerySAT consistently outperforms traditional and baseline neural network models across various SAT tasks, including complex challenges like 3-SAT and SHA-1 preimage attacks. These results not only demonstrate the effectiveness of integrating query mechanisms into neural solvers but also suggest potential for further enhancements in broader applications within EDA.

However, despite its strengths, the approach encapsulates inherent limitations, particularly regarding the scalability and generalization of the model to larger, more diverse SAT instances. The reliance on recurrent steps and the complexity of the unsupervised loss function might pose challenges in operational environments where computational resources and time are constraints. Additionally, the adaptation of such a model in a real-world setting would require extensive tuning and validation to ensure reliability and accuracy, particularly in high-stakes applications like hardware verification.

Graph-Q-SAT represented in [6] with a novel approach where the traditional SAT solving process, especially the branching heuristic, is enhanced through a Q-learning based algorithm utilizing graph neural networks (GNNs). This model stands out by training on standard MiniSat, but it significantly deviates by using a reinforcement learning framework to optimize the branching decisions dynamically, based on the state of the problem at any point in the solving process.

One of the most commendable features of Graph-Q-SAT is its ability to quickly adapt its strategy based on the problem instance, leading to a 2-3 times reduction in the number of iterations required to solve SAT problems. This efficiency not only speeds up the solving process but also exhibits strong generalization capabilities. The model was effective across various problem sizes and types, from SAT to unsatisfiable (unSAT) instances, demonstrating its robustness in different scenarios.

Graph-Q-SAT's training method is particularly notable for its data efficiency, requiring no extensive dataset preparation or complex feature engineering, which simplifies its integration and scalability in practical settings. Furthermore, it shows promising zero-shot transfer capabilities, performing well on problem types not encountered during training. This ability to generalize from limited initial data points to a broader array of problems is a significant step forward for machine learning applications in formal verification.

However, despite these strengths, the application of Graph-Q-SAT in an industrial context would require further development. The primary challenge remains its scalability and the computational overhead associated with running graph neural networks within the iterative solving process. There is also a need for more comprehensive testing and refinement to ensure reliability and consistency across even more diverse and complex SAT instances.

the Reference [7] presented by Emre Yolcu and Barnabás Póczos shows a method that further refines the capabilities of machine learning in this area. This research employs a deep reinforcement learning framework alongside a graph neural network to develop a variable selection heuristic within a stochastic local search algorithm, specifically designed for SAT solving.

The innovative aspect of their approach lies in the use of a graph neural network to enhance the heuristic decision-making in the SLS algorithm, focusing on learning optimized heuristics for various classes of SAT problems from scratch. This method allows for a more efficient search process by reducing the steps required to find satisfying assignments, thereby potentially shortening the verification cycles in electronic design processes.

This study highlights the effectiveness of employing graph neural networks to capture and utilize the intricate dependencies within SAT problems. The authors demonstrate that the learned heuristics can significantly outperform generic heuristics, especially in terms of the number of search steps required, although at a higher computational cost. The primary advantage is the tailored approach to different problem classes, which allows for a more nuanced and effective search strategy compared to one-size-fits-all heuristics.

However, the scalability of this method poses a challenge, as the computational expense grows with the complexity of the problems. This limitation may restrict its practical application to smaller or less complex instances unless further optimizations can be made. Additionally, the generalizability of the learned heuristics across different types of SAT problems without retraining remains an open question, potentially limiting the approach's flexibility.

4.2.2 Model Checking

Model checking is a critical technique in formal verification, designed to prove the correctness of design models relative to a formal specification using exhaustive state-space exploration in a strict mathematical way . It checks if a system model conforms to its specifications and can detect errors in systems with infinite states. The properties to be verified are specified in a formal language, typically temporal logic, such as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL). Given its complexity and resource intensity, advancements in integrating machine learning techniques to optimize model checking processes are pivotal.

The Reference [8] proposed by Elmandouh and Wassal explores the integration of supervised machine learning techniques specifically for the orchestration of formal verification engines in RTL designs. The authors leverage multi-class classification machine learning to effectively select the most suitable formal engines based on past performance data, which includes a vast repository of 16,500 formal verification runs.

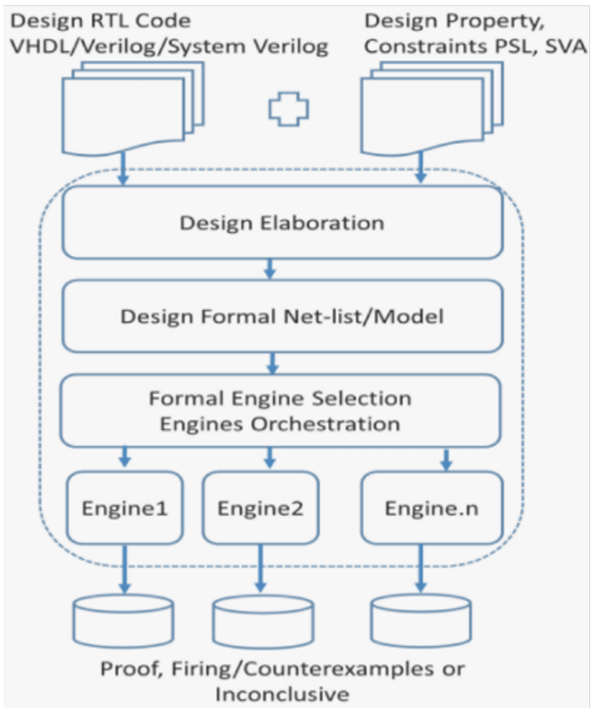

Figure 4.3: Multi-engines Formal Verification orchestrating Flow

This innovative approach addresses the complexity and diversity of modern hardware design by orchestrating a set of state-of-the-art formal verification algorithms as shows in Figure 4.3, allowing for efficient scheduling that minimizes the time consumed for verifying individual design properties. By predicting which formal engines are most likely to successfully verify a given property, the model ensures that verification tasks are not only quicker but also more accurate, with an impressive accuracy of 88% in predicting the appropriate formal engines for new designs. However, the scalability and the heavy reliance on historical data could be seen as potential limitations. While the system shows significant improvements in runtime, up to 59%, the generalizability of the approach to different or evolving hardware designs, where past data might not adequately represent future verification scenarios, remains a challenge.

Another Reference [9] introduces a novel approach utilizing machine learning to streamline the process of Linear Temporal Logic (LTL) model checking. This paper explores the potential of binary classification methods to predict the outcomes of LTL model checking, thereby reducing the computational complexity traditionally associated with this verification process.

The approach centers around training a binary classifier to predict whether a given model satisfies the specified LTL properties before executing a full model checking process. Within which they've used machine learning methods such as Boosted Tree(BT) Random Forest(RF) Decision Tree(DT) and Logistic Regression(LR) to make the result comparasion. This preliminary prediction aims to filter out the models that are likely to fail the LTL properties, thus potentially reducing the number of model checks needed. The classifier is trained on a dataset generated from historical model checking runs, which include features extracted from the models and their LTL specifications.

A significant advantage of this approach is its potential to decrease the computational resources required for model checking by preemptively identifying models that are likely to violate the LTL properties, thereby allowing computational efforts to be focused on more promising models. This can lead to faster verification cycles in electronic design automation, where model checking is a critical component of the verification process.

However, the approach also exhibits limitations. The accuracy of the binary classifier heavily influences the effectiveness of the entire model checking process. If the classifier's predictions are inaccurate, it could either miss some critical errors by falsely predicting compliance or waste resources by incorrectly flagging compliant models. Additionally, the training process requires a substantial dataset of prior model checks, which might not be readily available for all types of systems or might not represent future systems accurately, potentially limiting the generalizability of the model.

Finally the Reference [10] which focused on solving model-checking issue either introduces an innovative use of Generative Adversarial Networks (GANs) to automate the generation of verification properties for model checking, specifically targeting the generation of Computational Tree Logic (CTL) properties. This method represents a significant advancement in the domain of formal verification, where the manual generation of properties is often complex and time-consuming.

The core idea of this approach is to employ GANs, where a generator network attempts to produce realistic CTL formulas, and a discriminator network evaluates the realism of these formulas. The generator is trained on a dataset of existing verification properties, learning to replicate and innovate on the CTL property structure. The discriminator helps refine the quality of generated properties by filtering out those that do not adhere to CTL grammar rules.

One of the key strengths of this method is its potential to significantly reduce the manual effort involved in specifying verification properties, which can be a bottleneck in the model checking process. Additionally, the system's ability to learn and adapt to produce new properties that have not been explicitly taught offers a way to discover potentially overlooked verification scenarios.

4.2.3 Assertion Estimation

In another hand, the Reference [11] authored by Aditi and Michael S. Hsiao, explores an innovative approach combining rule-based systems and machine learning to generate SystemVerilog assertions directly from natural language specifications. This work addresses the complexity of translating natural language into formal verification code, a process typically prone to human error and requiring considerable manual effort.

This hybrid method employs both rule-based formal analysis and machine learning techniques, specifically utilizing models like GPT-3's Curie and DaVinci engines. The rule-based component parses natural language to understand its grammatical structure, identifying key elements such as signal names, values, and logical conditions. Machine learning, on the other hand, assists by generating the final assertions based on extracted information, aiming to enhance accuracy and handle more complex language structures that purely rule-based systems might struggle with.

One of the main advantages of this approach is its ability to significantly automate the assertion generation process, potentially decreasing the verification cycle time and reducing the likelihood of human error. This is particularly valuable in the EDA industry where verification can consume a substantial portion of the development cycle. Moreover, the method includes a validation step that cross-references the generated assertions with those produced by other machine learning models to ensure accuracy, thereby minimizing the need for costly manual validation.

Stage of EDA	Purpose	ML method
Formal Verification	Boolean SATisfiability	Gated-GCN [4] Recurrent-GNN [5] DQN(RL)+GNN [6] RL+GNN [7]
	Model Checking	Logistic Regression/MLP/Decision Tree [8] Boost Tree/Random Forest/Decision Tree/ Logistic Regression [9] GAN [10]
	Assertion Generation	GPT-3/OPENAI CODEX [11]
	Runtime Estimation	Ridge-Regression/Lasso-Regression [12]

Table 4.2: Summary of ML methods applied in Formal Verification

4.2.4 Runtime Estimation

he Reference [12] authored by Eman El Mandouh and Amr G. Wassal tackles the significant challenge of predicting the cost and runtime required for formal verification in hardware design processes. The authors utilize regression-based machine learning techniques to construct an estimation model that predicts the formal verification runtime based on attributes of RTL designs.

Their approach employs multiple regression techniques, including Ridge and Lasso regression, to handle the bias-variance trade-off and perform feature selection respectively. These methods are particularly suited to handle overfitting and select the most significant features from the dataset which includes 10,000 formal verification runs. The primary advantage of using these regression techniques is their ability to provide quantitative insights into which design features significantly impact the verification process's duration and complexity, thus allowing for better planning and resource allocation in hardware design projects.

Besides all above, this method also presents limitations, primarily the dependency on the availability and quality of historical data. Moreover, the regression models could struggle with non-linear relationships unless appropriately transformed or parameterized, which can complicate the model training and selection process.

The reviewed literature explicit the integration of machine learning with formal verification in electronic design automation. Highlighting significant advancements in automating and optimizing hardware verification processes. Across nine studies, various machine learning approaches, including neural networks, regression models, and reinforcement learning, are utilized to automate and optimize complex verification tasks. These techniques range from generating SystemVerilog assertions from natural language to predicting formal verification costs and enhancing solver efficiency. Despite their potential, challenges such as data dependency, model overfitting, and generalizability still exist. However, the collective findings suggest a promising shift towards more intelligent, adaptive verification tools that could significantly reduce verification times and improve the reliability of hardware designs.

4.3 ML for Logic Synthesis

In the stage of logic synthesis for electronic design automation, machine learning technologies are increasingly being implemented to enhance power estimation, QoR (quality of result) and the overall design process of converting higher-level descriptions of digital circuits into gate-level descriptions. The integration of machine learning approaches enables more efficient handling of the complexities and constraints typical in modern circuit design, such as power, timing, and area optimizations.

Machine learning methods, primarily deep learning and reinforcement learning, are applied to predict outcomes and guide the synthesis process. For example, deep learning models can be trained on historical synthesis data to predict the quality of synthesis outcomes based on various design choices. This predictive capability allows designers to evaluate the potential impacts of different synthesis strategies without fully implementing each one, thereby saving significant time and resources.

4.3.1 Power estimation

In the evolving field of EDA, particularly within the logic synthesis stage, the utilization of machine learning to estimate power consumption has emerged as a significant area of research. Two notable contributions in this area are the studies [13] and [14]. Both papers explore the application of graph neural networks (GNNs) but focus on different aspects and methods within the power estimation process, reflecting a trend towards more data-driven, intelligent approaches to handling the complexities of modern integrated circuit design.

GRANNITE focuses on leveraging transferable learning within GNN frameworks to estimate power consumption across different semiconductor technologies without the need for retraining. This approach is significant as it addresses the challenge of model obsolescence with changes in technology, making the power estimation process more adaptable and cost-effective. The method involves training a model on a specific technology node and then applying it to different nodes, highlighting its robustness and flexibility. The GNN model captures intricate dependencies within the chip layout and operational parameters, offering predictions that adapt to various configurations and usage scenarios.

GRASPE, on the other hand, delves into post-synthesis power estimation from Register Transfer Level

(RTL) designs using a graph representation learning approach. This technique is particularly geared towards providing high accuracy in power estimation by converting RTL designs into a graph format where nodes and edges represent components and their interactions, respectively. The paper underscores the accuracy of this method in reflecting the real-world power usage of synthesized circuits, helping designers make more informed decisions about power efficiency during the early stages of design.

Both papers utilize graph-based data structures to represent the intricate interactions and dependencies of circuit elements, which is a natural fit for capturing the complex relationships inherent in integrated circuits. GNNs are particularly suited to this task because they excel in mining relational data for patterns that traditional neural network architectures might miss.

However, these approaches are not without limitations. The success of GRANNITE's transferable model depends heavily on the similarity between the technology nodes. If the nodes are too divergent in terms of design and operational characteristics, the model's effectiveness may decrease. GRASPE, while accurate, requires a comprehensive and correctly labeled graph representation of RTL designs, which can be a resource-intensive process, potentially limiting its applicability in rapid, iterative design cycles.

By comparing these two methods, it's clear that while both leverage the strengths of GNNs, they apply them at different stages of the design and synthesis process and with slightly different aims—GRANNITE for transferability across technologies and GRASPE for immediate post-synthesis estimation accuracy. Such distinctions are crucial for understanding the scope and potential deployment scenarios of each method within the logic synthesis workflow. Subsequently, GRASPE was evaluated with the state-of-the-art GRANNITE for inference latency and average power estimation and demonstrate an average improvement of 3.985X and 1.28%, respectively.

Stage of EDA	Purpose	ML method
Logic Synthesis	Power Estimation	GNN [13] [14]
	QoR improvement	DNN [15] Recurrent-NN [16]
	QoR(tuning DSE automatically	RL [17]
	QoR(choosing the best oprimizer for different parts of circuits	DNN [18]
	QoR(choosing high-quality synthesis flow)	CNN [19]
	QoR(improving synthesis optimization algorithm)	RL [20]

Table 4.3: Summary of ML methods applied in Logic Synthesis

4.3.2 QoR Improvement

The Reference [15]introduces an innovative approach that integrates deep learning with approximate computing and low-power design to optimize logic synthesis. This work is particularly significant as it aims to address the Quality of Results (QoR) challenge in logic synthesis by efficiently minimizing power consumption while managing a predetermined error rate.

Deep-PowerX employs a deep neural network (DNN) to predict the error rates at the primary outputs of circuits when specific components of the netlist are approximated. This methodology allows the framework to replace or remove gates strategically to reduce dynamic power consumption and, secondarily, the area. One of the standout features of Deep-PowerX is its ability to reduce the time complexity of standard approximate logic synthesis from exponential to linear, making it highly efficient in terms of runtime.

The framework operates by using the trained DNN to guide the approximate logic synthesis engine, proposing replacements for each node in the circuit based on the error rates predicted by the model. If the predicted error exceeds the user-defined threshold, the replacement is aborted, ensuring that the circuit maintains acceptable accuracy levels. This process not only promises significant reductions in power and area but also maintains a balance between performance and accuracy, a critical requirement in many modern electronic applications where some degree of computational error can be tolerated for gains in efficiency.

In another Reference [16] Cunxi Yu and Wang Zhou present a sophisticated machine learning framework to predict Quality of Result (QoR) metrics, such as delay and area, for synthesis flows across different semiconductor technologies. Their methodology utilizes Long Short-Term Memory (LSTM) networks combined with transfer learning to forecast the impacts of various synthesis decisions on unseen design flows.

The core innovation of their approach lies in the application of LSTM networks to model the sequential decision-making process inherent in logic synthesis. The flows are represented in a novel "timed-model" format, which encodes the sequence and timing of synthesis operations into a two-dimensional matrix. This representation feeds into an LSTM model that predicts QoR outcomes based on past data collected from synthesis runs.

One of the significant advantages of their approach is the substantial reduction in the required size of the training dataset, facilitated by transfer learning. This method enables the LSTM model, initially trained on a dataset from 14nm technology, to adapt effectively to different technologies such as 7nm with minimal additional

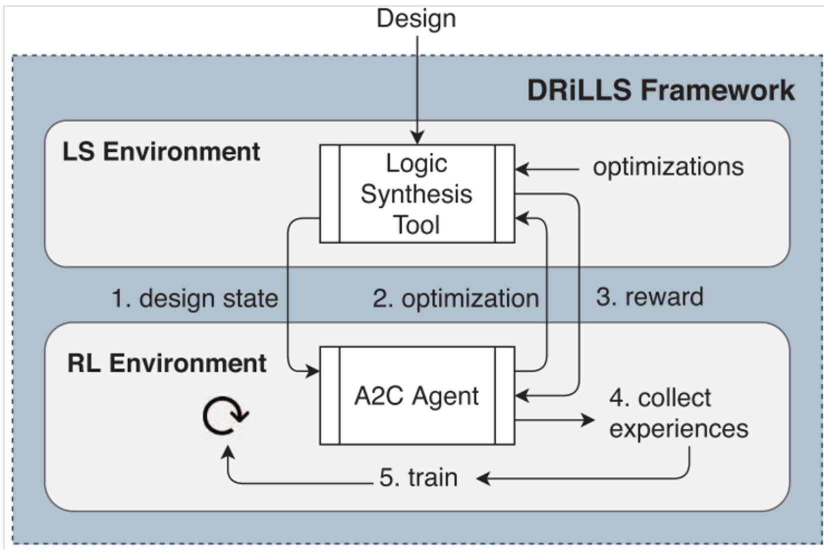

Figure 4.4: Drills Framework with A2C Agent

training data. The model demonstrates impressive accuracy, with a Mean Absolute Percentage Error (MAPE) of 3.7% across technologies, indicating its robustness and the efficacy of the transfer learning strategy.

By improving the QoR indirectly, the Reference [17] authored by Abdelrahman Hosny, Soheil Hashemi, Mohamed Shalan, and Sherief Reda explore the utilization of reinforcement learning to automate and optimize the logic synthesis process, a crucial aspect in improving the QoR. The focus is on optimizing logic synthesis to minimize area while meeting specific timing constraints, which are common goals in the synthesis of digital circuits.

The DRiLLS framework employs an Advantage Actor Critic (A2C) model as shows in Figure 4.4, a sophisticated type of reinforcement learning algorithm that combines the benefits of policy-based and value-based methods. This model operates in a simulated environment where the sequential and combinatorial nature of logic synthesis can be modeled as a decision-making game with clear rules and objectives. The state of the design at any step is represented as a feature set derived from the And-Inverter Graph (AIG), which characterizes the circuit. The reward function is carefully formulated to balance the need for area reduction with the adherence to delay constraints, providing a direct incentive for the learning agent to find optimal synthesis transformations. One of the key innovations of DRiLLS is its ability to operate autonomously, reducing the reliance on human expertise traditionally required to tune the synthesis process. This is particularly beneficial given the increasing complexity of modern integrated circuits and the vast design space that needs to be explored to find optimal

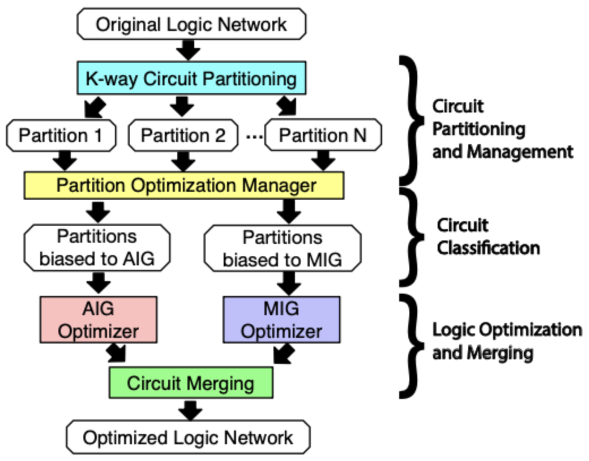

Figure 4.5: LSOracle optimization Framework with k-way partitioning and Classification used to allocate different partitions with different optimization algorithms

solutions. The use of A2C allows the system to effectively learn from past actions and improve its strategy over time, which is demonstrated through comparative benchmarks where DRiLLS consistently outperforms traditional and other machine learning-enhanced methodologies.

Another Reference [18] which utilized both And-Inverter Graph (AIG) and Majority-Inverter Graph (MIG) optimizers either presents a novel approach to logic synthesis by manipulating a heterogeneous framework that combines different types of optimizers to improve the Quality of Results (QoR) for various circuit designs. This innovative framework, LSOracle, leverages a Deep Neural Network (DNN) to intelligently decide which optimizer is best suited for different segments of a circuit. This decision is based on the partitioning of the circuit into multiple segments, which are then individually optimized.

One of the standout features of LSOracle is its ability to adaptively choose the most appropriate optimization technique for each partitioned segment, making it highly efficient in handling complex, mixed-logic circuits typically found in modern Systems-on-Chip (SoC). The framework as shows in Figure 4.5applies k-way partitioning to divide the circuit into manageable parts and then employs the DNN to select the optimal logic optimizer for each part based on its characteristics. This approach not only improves the area-delay product but also enhances the overall performance and efficiency of the logic synthesis process. However, the effectiveness of LSOracle depends significantly on the accuracy of the DNN's predictions, which in turn relies on the quality and diversity of the training data used. Additionally, while LSOracle shows substantial improvements in handling mixed-logic circuits, the complexity of its setup and the computational resources required for running the DNN may pose challenges, especially in smaller or resource-constrained environments

To achieve better QoR with choosing the high-quality synthesis flow autonomously, the Reference [19] explores the application of convolutional neural networks (CNNs), improving the Quality of Results (QoR) in electronic design. This approach is significant as it targets the automation of a highly complex and traditionally manual process, potentially reducing the need for expert input and expediting design iterations.

This research utilizes a CNN to classify different synthesis flows into those that provide the best and worst QoRs, named angel-flows and devil-flows respectively shows as the Architecture 4.6. The CNN model operates by inputting high-level description language (HDL) and outputs optimized synthesis flows tailored to specific design objectives like delay reduction or area minimization. This method showcases the power of deep learning in handling the vast space of synthesis flows, which are often impractical to explore through human testing alone due to their complexity.

A key strength of this approach is its capacity to learn from data and make informed decisions without explicit programming for each specific task. By leveraging a CNN, the system can predict the most effective synthesis strategies from a substantial dataset of potential flows, making it highly scalable and adaptable to different IC design scenarios.

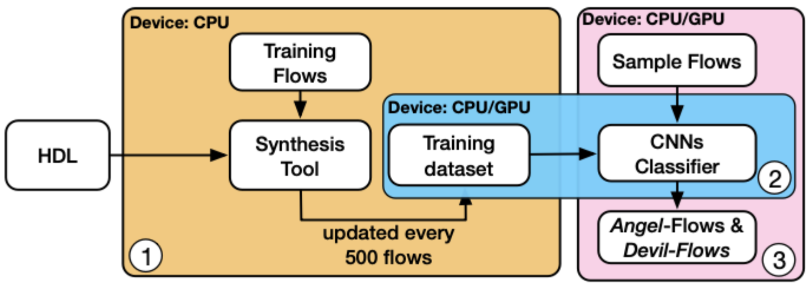

Figure 4.6: Automatically high-quality synthesis flows choosing architecture with angel-flow and devil-flow only

Finally, the Reference [20] authored by Winston Haaswijk and colleagues introduces an innovative approach to enhancing logic synthesis optimization algorithms using deep reinforcement learning. This study aims to automatically discover optimal configurations for logic networks by modeling the optimization process as a deterministic Markov decision process (MDP). By doing so, it leverages recent advancements in deep learning to automate and improve the Quality of Results (QoR) in logic synthesis without human intervention.

This approach is innovative in that it allows the model to autonomously generate optimization strategies without human intervention, learning to identify and execute the most effective transformations on logic networks. The system uses a graph convolution network (GCN) to evaluate and score potential moves within the optimization process, optimizing for size and depth of the logic circuits.

One of the key advantages of this method is its autonomy and ability to generalize from small to larger logic functions. The framework has demonstrated impressive improvements in optimization tasks, achieving up to 100% of potential improvement for 3-input functions and significant gains for more complex circuits as demonstrated in the MCNC benchmark case studies. This showcases the DRL model's capability to adapt and perform well across different scenarios and circuit complexities.

In conclusion, the integration of machine learning, specifically through graph neural networks, into power estimation for logic synthesis presents a promising advancement in electronic design. By automating and enhancing accuracy in power estimation, these methods support more efficient and sustainable hardware design practices, aligning with the industry's ongoing efforts to manage power consumption in the era of complex multi-functional devices. These studies not only push the boundaries of what's possible with current technology but also pave the way for future innovations in EDA. The rest 6 articles on machine learning applications in logic synthesis collectively demonstrate a shift towards automated and intelligent systems, significantly enhancing the quality of results (QoR) and optimizing the synthesis process. These studies leverage advanced techniques like graph neural networks and deep reinforcement learning to tackle complex optimization challenges in electronic design automation (EDA). While these methods automate and improve efficiency, they also face challenges such as scalability, computational demands, and the need for high-quality training data. Overall, the integration of machine learning into logic synthesis heralds a promising future for more efficient design automation tools capable of addressing the increasing complexity of integrated circuits.

Stage of EDA	Purpose	ML method
Placement	Minimizing Wire(Net)Length	GNN [21] GNN(GraphSAGE)+RL [22]
	Provide Placement-Guidance	GNN(GraphSAGE) [23]
	Policy Gradient Optimization	RL [24]
	Minimizing DRC Violations	CNN(VGG16) [25]
	Timing Prediction	Random Forest [26] Linear Regression/NN/Random Forest [27]
	Path-delay Prediction	Transformer Network [28]

Table 4.4: Summary of ML methods applied in Placement

4.4 ML for Placement

Once again, attributing to the rise in the need for semiconductor ICs, generating a layout from a netlist is essential in the IC design process. Physical design is converting a logical netlist or RTL into a physical layout. Most fabrication processes require design houses to use certain design libraries specific to their fabrication process. Generating these design layouts from the design netlist and other design files is complex and time-consuming. Chip placement is an important processes in the IC design flow. Finding the optimal floor plan and placement of a design is considered one of the most time-consuming and complex processes. Modern IC designs have numerous smaller IPs, macros, and modules that require multiple iterations of placement to figure out the optimal position for each instance. Most ICs do not have the most optimal placement simply because of the high number of placement permutations possible, even for small designs. This is exacerbated by larger designs, and a solution to find the best placement while using reasonable resources is a challenge. Multiple research teams have been invested in finding solutions to improve the placement.

4.4.1 Wirelength Minimization

The exploration of machine learning applications in the placement phase of VLSI design focuses on improving the optimization of wire and net lengths, crucial for enhancing the overall performance and efficiency of electronic circuits. Two notable papers in this domain employ advanced machine learning techniques, each bringing a unique approach to addressing the challenges associated with placement optimization. The first Reference [21] introduces a method utilizing graph attention networks (GATs) to predict net lengths before the actual placement occurs. This predictive model is designed to guide the placement tool by providing early insights into potential net length issues, allowing for preemptive adjustments. The use of GATs enables the model to focus on specific parts of the netlist that contribute most significantly to net length, dynamically adjusting its attention based on the structure of the netlist and the connections between components. This approach not only helps in predicting the net lengths more accurately but also aids in understanding how different elements within the netlist interact, potentially identifying critical areas that require special attention during placement. The strength of this method lies in its predictive accuracy and the ability to handle complex netlist graphs by learning which connections are

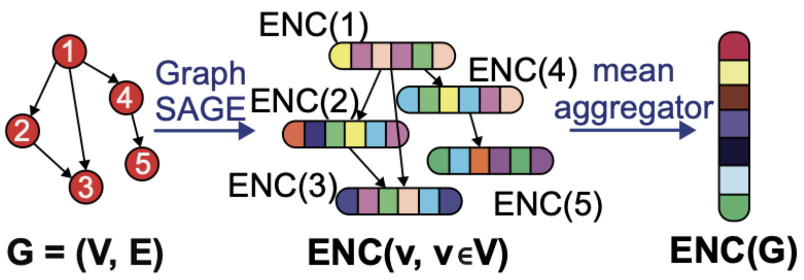

Figure 4.7: GraphSAGE (a GNN package used for graph embedding) In their experiments, 32 features were extracted firstly for each node in the graph. Next, the mean among all nodes for each feature were calculated. In the end, 32 features were obtained for the en- tire graph.

most influential in determining net length.

The second Reference [22] models the placement problem as a Markov decision process (MDP) where each action taken by the RL agent adjusts placement parameters that affect the overall wire length of the circuit. It takes a different approach by applying deep reinforcement learning (RL) to optimize placement parameters dynamically. This method leverages the capabilities of reinforcement learning to make sequential decisions, adapting the placement strategy based on real-time feedback and iteratively improving the placement to minimize wire lengths. The reinforcement learning model operates by learning from the environment it creates through each placement iteration, effectively learning from its actions to optimize both the placement process and its outcomes in terms of wire length and other crucial parameters like timing and power.

This approach benefits from the flexibility and adaptability of reinforcement learning, which can continually improve as it encounters new design scenarios. It also directly manipulates placement parameters, potentially leading to innovative placement strategies not considered in traditional methods. Based on the transformed graph and the initial node features defined for each instance, they leverage GraphSAGE 4.7, a variant of GNNs, to perform unsupervised node representation learning.The main challenges here include the need for significant computational resources to run numerous simulations for training and the potential difficulty in translating the learned strategies to physically realizable designs. Both approaches share a common goal of minimizing wire and net lengths but differ significantly in their methodologies. The GAT-based model focuses on predictive analytics, providing valuable foresight into potential placement outcomes, which can be used to make informed decisions early in the design process. On the other hand, the RL-based approach is more dynamic and adaptive, learning from ongoing processes and capable of adjusting strategies in real-time to find optimal solutions.

4.4.2 Performance Optimization

Provide Placement Guidance The Reference [23] presents a comprehensive approach to addressing the challenges of VLSI placement, specifically focusing on optimizing the placement process to minimize wire and net length. The methodology employs a framework known as PL-GNN that integrates unsupervised machine learning techniques with traditional VLSI design workflows to enhance placement guidance provided to commercial placers like Synopsys IC Compiler II (ICC2).

PL-GNN framework operates in two main stages as in Figure 4.8. Initially, it employs unsupervised node representation learning using graph neural networks (GraphSAGE 4.7) to analyze a netlist graph and extract features that accurately represent the logical affinity among different design instances. The graph-based representation is crucial as it allows the model to understand and predict how instances should be grouped to optimize specific design metrics such as wire length, congestion, and timing. After learning the node representations, the framework utilizes weighted K-means clustering to group instances into clusters, which are then used as guidance for the placement tool to achieve optimal placement.

One significant advantage of this method is its ability to automate parts of the placement process that traditionally require in-depth, design-specific knowledge, thereby reducing the reliance on experienced human designers. The use of GNNs enables the PL-GNN to handle complex netlist structures and optimize them effectively, leading to demonstrable improvements in wire length, power consumption, and timing, as shown in the results from commercial multi-core CPU designs.

However, the computational complexity of training and applying GNNs can be significant, especially for very large netlists typical in modern VLSI designs.

Overall, the PL-GNN framework represents a promising fusion of graph neural networks and traditional electronic design automation, providing a novel method that improves the efficiency and effectiveness of the VLSI placement process. This approach not only contributes to reducing the overall design time but also enhances the quality of the final chip layout, potentially setting a new standard for how machine learning can be integrated into EDA tools.

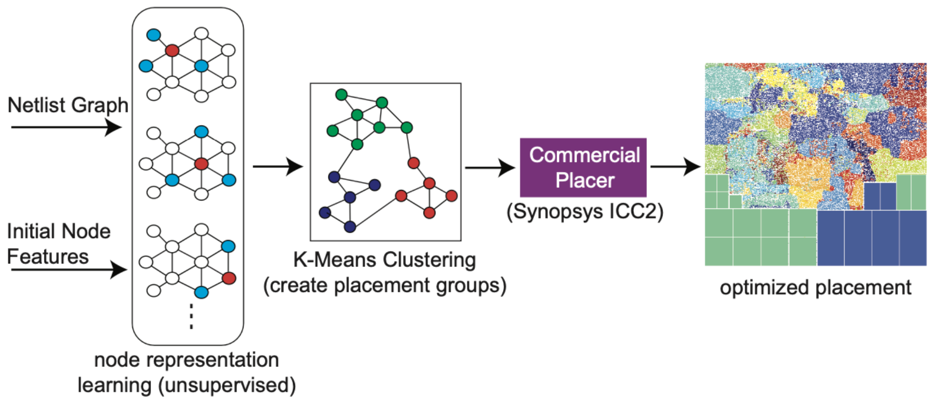
Netlist Graph
Initial Node Features
node representation learning (unsupervised)
K-Means Clustering (create placement groups)
Commercial Placer
(Synopsys ICC2)
optimized placement

Figure 4.8: PL-GNN (a transferable framework developed to provide accurate/automated placement guidance for any design)

Policy Gradient Optimization To address the placement optimization problem in VLSI design, the Reference [24] authored by Anna Goldie and Azalia Mirhoseini introduces a novel approach using deep reinforcement learning (RL). The focus is on improving the placement of nodes within a limited resource environment to optimize objectives such as power, performance, and area, under specific constraints.

The authors propose using policy gradient optimization, a method within deep RL, which involves an agent learning to make decisions in an environment that maximizes a cumulative reward. This approach stands out because it transcends traditional placement algorithms, which often rely on heuristic or deterministic methods, by incorporating a model that learns and adapts from the environment to find optimal solutions iteratively.

The core of this methodology is to formulate the placement problem as a Markov decision process (MDP) either, where the RL agent interacts with an environment represented by a state (the current placement of nodes), takes actions (adjustments to the placement), and receives feedback in the form of rewards (measures of placement efficiency based on predefined metrics). The reinforcement learning model, which is essentially a deep neural network, leverages the benefits of policy gradient methods like REINFORCE and Proximal Policy Optimization (PPO) to navigate the placement landscape effectively.

Minimizing DRC Violations To minimize design rule violations (DRVs) with improving macro placement specifically, the Reference [25] offers an innovative approach by combining deep learning, a Convolutional Neural Network(CNN) within the macro placement process to predict routability issues before they occur, enabling more informed and effective placement decisions.

The methodology centers around a CNN-based predictor which is embedded directly within a macro placer. This setup allows the placer to evaluate potential macro layouts based on the predicted number of DRVs, which is a measure of how likely a given placement is to meet stringent design rules without extensive modifications. The CNN model is trained on data representing different macro placements, where the model learns to correlate placement patterns with routability outcomes. This predictive capability is leveraged through a simulated annealing optimization process, aimed at finding a macro placement with minimized potential for DRVs. While stacking the three 2-dimension feature maps (macro density map/pin density map/connectivity map) to a 3-dimension input tensor the normalization is applied to each map to keep them unanimous.

The integration of the CNN allows for a significant enhancement in the macro placement process by enabling predictions that guide the placement tool away from configurations likely to result in high DRVs. This proactive approach helps in refining design layouts much earlier in the design process, potentially reducing the time and cost associated with later-stage corrections.

However, this paper does have limitations, its results shared only with less 6 macros which is not realistic case, usually there would much more than that. Besides it not includes any optimizations of Physical design stage, this ignores a really significant factor that could affect DRC seriously. At last, any parameters for wire-length and timing optimizations are not included either.

Path-delay Prediction The Reference [28] authored by Peng Cao, Guoqing He, and Tai Yang, introduces an advanced machine learning model aimed at enhancing pre-routing path delay predictions in VLSI design. This study leverages a transformer network combined with a residual model to predict path delays more accurately before routing, addressing the gap between placement and routing stages that typically complicates the physical design process due to timing mismatches.

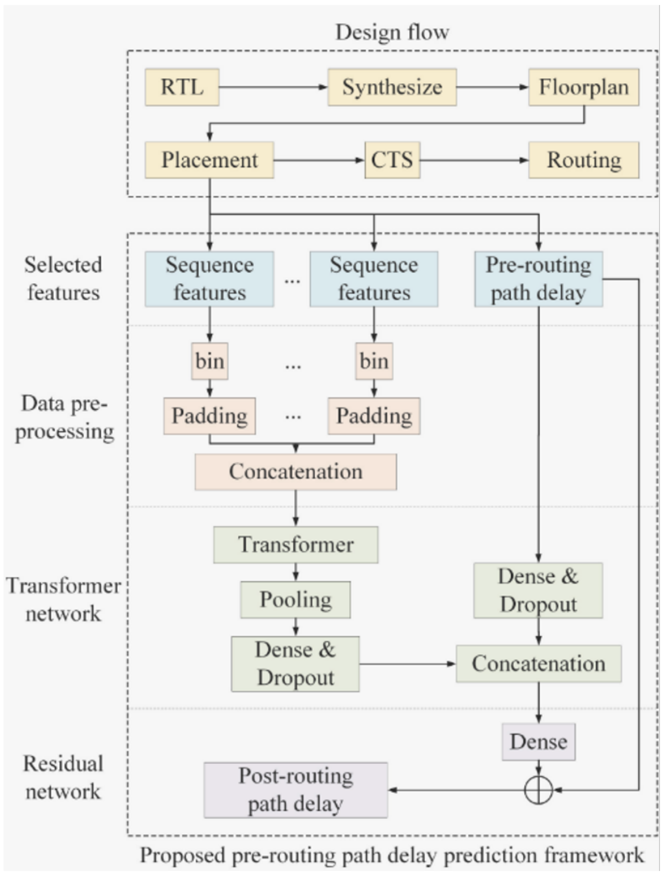

Figure 4.9: Transformer network and residual network-based pre-routing path-delay prediction framework

The core methodology involves using transformer networks to handle feature sequences extracted from each cell along the timing path as in Figure 4.9, allowing the model to grasp complex architecture and temporal dynamics within the circuit's layout. This is paired with a residual model that helps adjust the predictions by calibrating the initial estimates based on the discrepancies observed between the pre-routing predictions and actual post-routing outcomes.

The main advantage of this approach is its precision in predicting path delays with significantly reduced error rates (limited within 1.3% to 3.0%) and high correlation coefficients (above 0.995), which outperforms previous learning-based models. Such high accuracy is critical for avoiding costly design iterations and over-design, leading to more efficient power, area, and performance trade-offs. Furthermore, the model achieves a substantial speedup in prediction times compared to traditional methods and other competitive learning models, showcasing its potential for practical application in accelerating the VLSI design process.

4.4.3 Timing Prediction

Timing closure is a critical but effort-taking task in VLSI designs. In placement stage, a fast and accurate net delay estimator is highly desirable to guide the timing optimization prior to routing, and thus reduce the timing pessimism and shorten the design turn-around time. A fast net-delay timing predictor Can significantly reduce Time-pessimism and design Turn-around time as in Figure 4.10!

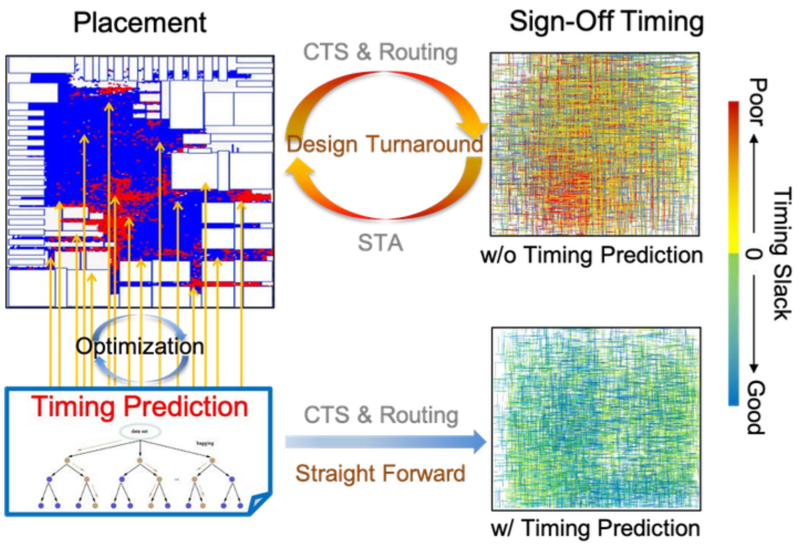

Figure 4.10: Applying machine learning-based timing model at the placement stage can improve timing prediction and de- liver better design performance

The two studies dedicate into enhancing the accuracy of timing predictions during the placement stage using machine learning techniques. Both aim to refine pre-routing timing estimations but approach the problem differently to minimize the necessity for extensive post-routing adjustments and to optimize overall design efficiency.

The first Reference [26] introduces a method that integrates a look-ahead RC network for feature extraction shows in Figure 4.11. This novel approach provides accurate early-stage timing predictions, thus reducing the requirement for conservative design margins that typically lead to overdesign. By predicting endpoint slacks and critical paths with high accuracy, the model shows significant improvement over traditional methods, achieving high correlation with post-routing sign-off timing results. Within the experiments, the Random Forest machine learning technology performs the best sicne it offered the best prediction accuracy.

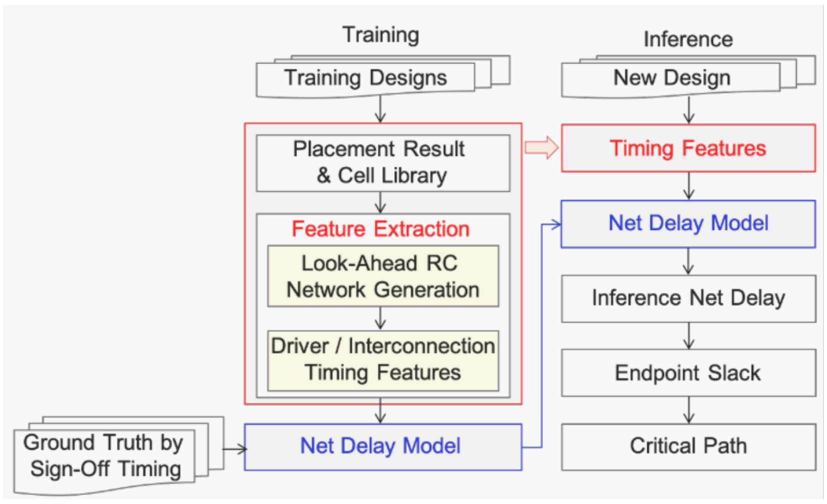

Figure 4.11: Framework of LaRC-Timer (look-Ahead RC network used to extract features as the input of placement in the meanwhile the circuit graph are traversed in topology order to compute the arrival time of each node

The second Reference [27] focuses on reducing the typically pessimistic pre-routing predictions that can lead to inefficient chip resource use. It employs machine learning models shows in Table 4.5 to predict individual net delays and incorporates this information into static timing analysis. The approach effectively reduces the false positive rate of timing violation reports and decreases the mean squared error compared to conventional estimation tools, enhancing the accuracy of pre-routing timing predictions.

	Characteristic	Setups	Flaws	Adavantages
Lasso	Linear Regression Algorithm	Scikit-Learn Libarary	Too simple	Really Simple to implement
Neural Network	Non-Linear Activation Function	Keras	Lack of interpretability	Easy to customize loss-function
Random Forest	Decision(if-then-else)Trees based Model	Scikit-Learn Library	Rarely support customization of loss-function	Better interpretability
K-Nearest Neighbor(KNN)	Non-Parametric Tech	Scikit-Learn Libarary	Lack of interpretability	Easy to customize loss-function
Artificial Neural Network(ANN)	Model Non-Linear Relationships	Scikit-Learn Libarary	Chanllenging on training	Flexible in learning various features

Table 4.5: The relevant Machine Learning models used in References and the flaws and advantages separately

Both methods leverage machine learning to not only improve the accuracy of timing predictions at the placement stage but also to reduce the need for subsequent design iterations. They share the challenge of requiring extensive training data to achieve high levels of accuracy, depending on the representativeness of this data for performance across various design scenarios. Additionally, the computational complexity of training and implementing these models poses practical challenges, particularly for very large or complex designs. Based on their tests, the Lasso and Random Forest were implemented using Scikit-Lear library while the Neural Network was implemented using Keras, with another commercial prediction tool tested in the experiments, the Random Forest still performed the best.

In summary, these studies highlight the potential of machine learning to significantly impact the timing prediction process in VLSI design especially with the Random Forest, providing more precise and less conservative estimates that can lead to more efficient and cost-effective chip designs. As the industry continues to advance, integrating such sophisticated methodologies will be crucial in meeting the increasing challenges of electronic design automation.

4.5 ML for Routing

The routing step in EDA design is crucial for determining the ultimate performance and manufacturability of integrated circuits. This phase involves the precise placement of wires connecting various components while adhering to stringent design rules and avoiding congestion. As technology nodes shrink and circuit complexity increases, traditional routing methods are challenged, necessitating more sophisticated solutions. Recently, there has been a significant shift towards incorporating machine learning techniques to address these challenges, particularly in predicting design rule violations (DRC), congestion issues, and overall routability.

The integration of machine learning offers a promising avenue to preemptively identify potential problems that could arise during the routing process, such as DRC violations and areas of high congestion, which traditionally were only detectable after the routing attempt had been made. By predicting these issues, designers can make informed decisions early in the design cycle, potentially saving substantial time and resources in later stages. Moreover, advancements in machine learning have also enhanced routability prediction, enabling more accurate modeling of complex interactions and dependencies within the routing layer. The following references aime to these two purpose in routing stage.

Stage of EDA	Purpose	ML method
Routing	DRC Prediction(Congestion Prediction)	Linear Regression/ANN/RF/K-nearest Neighbor [29] GAT [30] CNN [31] GNN+U-Net [32]
	Routability Prediction	CNN [33] FCN [34] Conditional-GAN [35] FCN [36] Heterogeneous-GNN [37]

Table 4.6: Summary of ML methods applied in Routing

4.5.1 DRC/Congestion Prediction

The Reference [29] presents a novel approach to addressing congestion issues during the routing stage of FPGA design. The authors utilize machine learning techniques to predict areas of congestion that can result from the placement and routing phases, aiming to provide earlier insights that can guide design decisions to avoid costly redesigns and iterations.

This study adopts a supervised learning approach, employing a Random Forest model, to predict congestion based on features extracted from the design prior to routing. Features include various properties of the FPGA architecture and the specifics of the initial placement, such as the number and type of logic blocks used, their connectivity, and preliminary wire length estimates. The choice of Random Forest is motivated by its ability to handle high-dimensional data and its robustness in the presence of noise and overfitting, making it well-suited for the complex and varied data encountered in FPGA design.

The use of machine learning, specifically Random Forests, offers several advantages. It allows for the modeling of non-linear relationships between design features and congestion, which are often not captured by traditional analytic models. The predictive model can be trained on historical data, improving its accuracy and reliability over time as it learns from a wider array of design scenarios. This predictive capability enables designers to anticipate and mitigate potential congestion issues before they manifest during routing, potentially saving significant time and resources.

Another Reference [30] represents a significant advance in the use of machine learning to predict routing congestion in EDA design, specifically focusing on the early prediction of logic-induced routing congestion from gate-level netlists before the placement and routing stages. This approach is particularly valuable as it can preemptively inform design adjustments to alleviate potential congestion issues, enhancing both the efficiency and quality of the resulting chip designs.

The core methodology of the study leverages graph convolutional neural networks (GCNs), specifically Graph Attention Networks (GATs), to analyze the intricate connectivity and interdependencies within the netlist graph. By focusing on the local logic structure, which significantly influences congestion at the lower metal layers, the model can predict congestion hotspots with remarkable accuracy. The use of GATs allows for adaptive learning of the importance of each connection in the graph, providing a nuanced understanding of how different

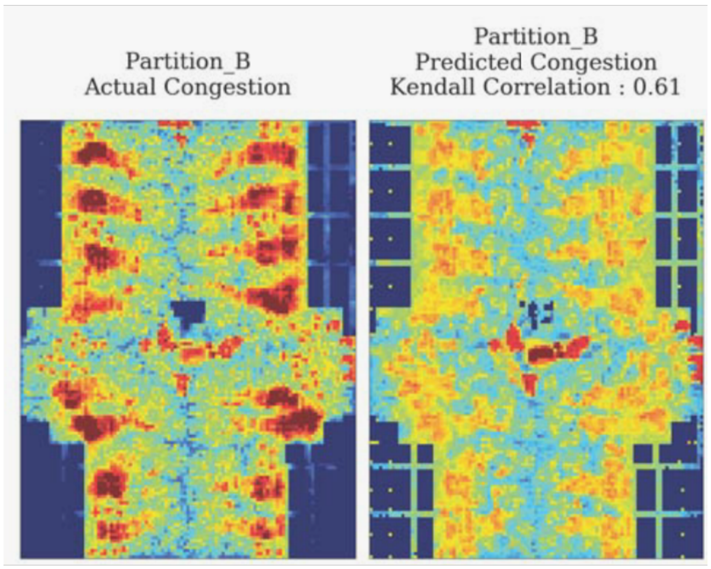

Figure 4.12: Best Partition of experiment result (left : Detail Routed Congestion; right : GAT Congestion Prediction; Both are in Lower half of Metal stack

elements contribute to overall congestion.

The predictive power of this model is underscored by its ability to achieve a 29% increase in the Kendall ranking correlation score over previous congestion prediction metrics, which did not utilize placement information. As shows in Figure 4.12, partition_B performs the best with Kendall Correlation of 0.61. This enhancement is even more pronounced when focusing on the lower metal layers, where the model's advantage over traditional metrics increases to 75%. Additionally, the speed of the model is noteworthy; it can predict congestion in a matter of seconds for circuits with millions of cells, a significant improvement over the hours required by other methods. In the other hand, the Reference [31] tackles the increasingly complex challenge of design rule checking (DRC) violations at advanced semiconductor process nodes. It introduces a novel convolutional neural network (CNN) architecture, J-Net, specifically customized to predict DRC hotspots effectively without the need for global routing information, which is typically a time-consuming process.

The methodology centers on addressing the mixed resolution of input data, where high-resolution pin shape features and lower-resolution layout features must be processed together efficiently. The J-Net architecture shows in Figure 4.13 is tailored to manage these varying resolutions by feeding different input channels into the encoder path at appropriate levels, allowing the network to handle the complexity of advanced node design layouts. This approach helps the model predict DRC hotspots by focusing on pin accessibility, a critical issue at sub-10nm

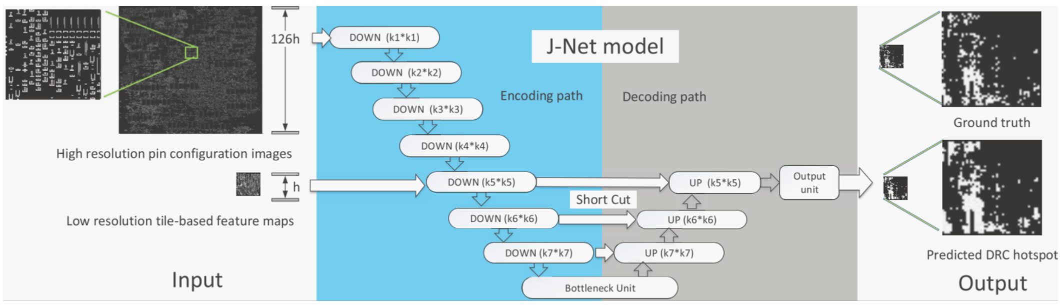

Figure 4.13: J-Net Architecture (different levels of input side deal with different resolutions; the number of levels of encoder is more than the number of levels of decoder so that the output resolutions are definitely less than the input

scales.

The key advantage of using J-Net is its ability to perform DRC hotspot prediction early in the design process, potentially avoiding the need for numerous iterations between placement and routing. This can lead to significant time savings and cost reductions in the chip design cycle. The study demonstrated that J-Net could achieve higher true positive rates compared to previous models, indicating its effectiveness in identifying potential DRC issues more accurately. Nevertheless, despite all the advantages of J-Net there's still flaws like huge time consuming that heavily influence the whole efficiency. Furthermore, as aimed to fix the problem of DRC Prediction, it's totally not considering the aspect of pin accessibility. Taking these two major factors into consideration, the Reference [32] introduces a sophisticated methodology to tackle the challenge of DRC hotspot prediction in EDA design. This study highlights the integration of a Graph Neural Network (GNN) with a U-Net architecture to create a novel model that effectively predicts potential hotspots by considering pin accessibility and routing congestion simultaneously.

The approach called PGNN utilizes a pin proximity graph to model the spatial relationships and interactions between pins, capturing how various design elements might influence pin accessibility. This graph structure allows for detailed representation and manipulation of the physical characteristics of circuit design, which are crucial for predicting DRC hotspots. The GNN is employed to process this graph, focusing on learning the intricate dependencies between the pins based on their spatial and connectivity features.

Simultaneously, as shows in the Figure 4.14the U-Net architecture is used to handle grid-based features that represent routing congestion data. This combination allows the model to integrate detailed pin-level information with broader congestion patterns to predict hotspots more accurately. By concatenating the outputs of the GNN

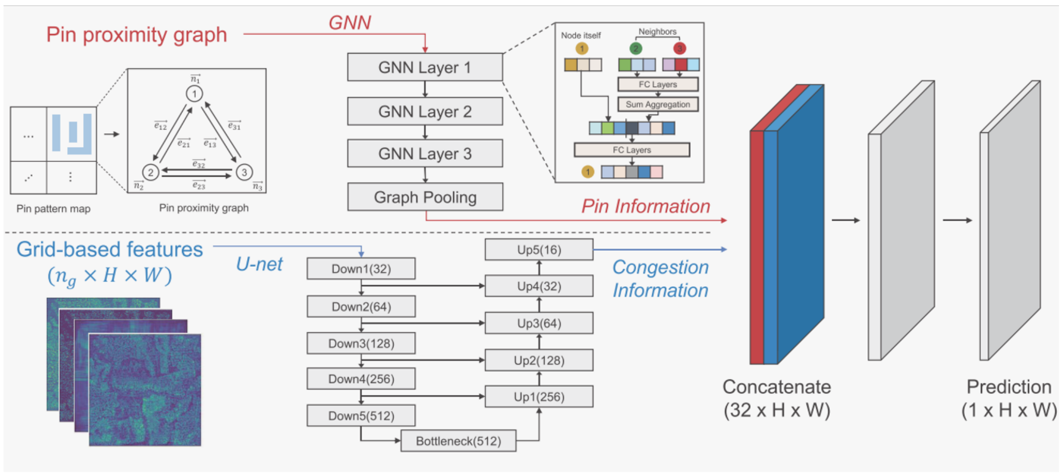

Figure 4.14: The architecture of PGNN including Pin proximity graph used for pin accessibility information and U-Net for congestion-information extraction

and U-Net, the model provides a comprehensive overview of potential DRC issues, leading to more reliable and actionable insights early in the design process.

One significant advantage of this method is the significant improvement in prediction accuracy and speed compared to traditional techniques. The fusion of GNN and U-Net allows for a deep understanding of both micro-level (pin accessibility) and macro-level (routing congestion) features, enhancing the model's predictive capabilities. However, the complexity of the model and the requirement for extensive training data can be challenging, necessitating substantial computational resources and robust training datasets to achieve optimal performance. As the last study which focus on solving DRC/Congestion Prediction problem, PGNN presents a much more advanced method to make improvement over training time(which only takes 3.8 hours while J-Net needs 31.7 hours), in the meanwhile the high-accuracy created by J-Net is maintained and even increased a little bit. This phenomenon is specially obvious on F1-score settings.

4.5.2 Routability Prediction

Furthermore, to predict routability based on the number of DRVs, the Reference [33] authored by Zhiyao Xie et al. explores the utilization of convolutional neural networks (CNNs) to predict routing challenges in EDA design, focusing on mixed-size designs which include large macros. The study emphasizes the significance of early routability prediction to proactively identify and resolve potential DRVs, which can be a cumbersome task at later stages.

RouteNet, the proposed CNN-based model, evaluates cell placement solutions for their overall routability or pinpoints specific locations prone to DRC hotspots. This method integrates features from various design stages into a trainable model that recognizes patterns indicative of routing issues. One of the main advantages of RouteNet is its speed and accuracy, achieving comparable results to traditional global routing methods but with significantly less computational time and overhead. RouteNet presents a significant advancement in using machine learning to predict routability issues, particularly for complex mixed-size projects where traditional methods may falter due to scale and intricacy.

The CNN model leverages the spatial features of chip layouts, treating them similarly to images, which allows for effective pattern recognition and generalization across different design scenarios. This approach not only enhances the prediction accuracy but also scales down the runtime, offering a pragmatic solution for early design stages.

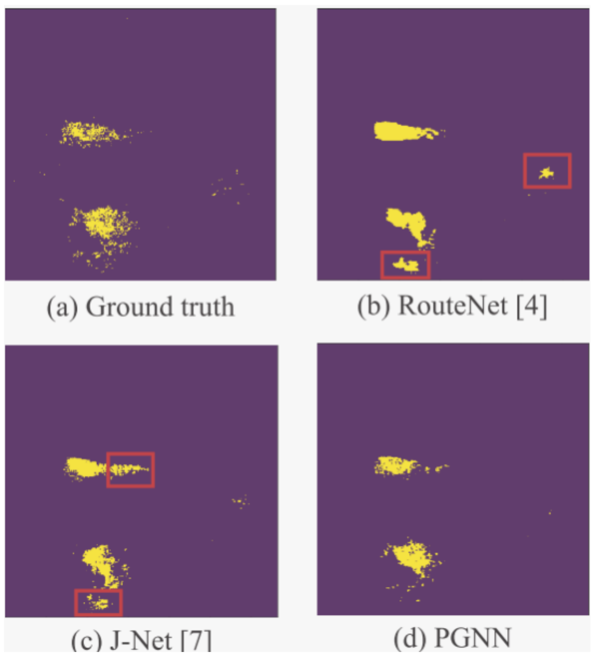

Figure 4.15: The Comparison result between Ground Truth with RouteNet, J-Net and PGNN

The comparison of framework RouteNet [33], J-Net [31] and PGNN [32] with Ground Truth shows in Figure 4.15. To addresses the crucial aspect of routability optimization, another Reference [34] proposed a deep learning-based plug-in named PROS.This plug-in integrates seamlessly into modern commercial EDA tools to predict and optimize routing congestion before detailed routing occurs, substantially reducing DRC violations.

PROS leverages a fully convolutional network (FCN) to predict global routing (GR) congestion using only data from the placement result, avoiding the overhead typically associated with feature preparation in conventional methods. The unique strength of PROS lies in its ability to operate with negligible runtime overhead, making it highly practical for integration into existing design workflows. The FCN model within PROS is specifically excellent at capturing spatial patterns related to congestion, which facilitates more accurate predictions and effective optimization of GR cost parameters to improve the overall routability.

However, the demonstrated efficacy of PROS in reducing DRC violations by 11.65% on average across tested industrial designs underscores its potential to significantly enhance the routability optimization process in advanced technology nodes.

Another Reference [35]introduces an advanced machine learning approach to enhance routing congestion prediction at the placement stage of large-scale FPGA design. By reformulating the prediction challenge into an image translation task, the study employs state-of-the-art generative adversarial networks (GANs), specifically designed for high-resolution image translation tasks, to produce routing congestion maps from features extracted during the placement stage.

The methodology utilizes a sophisticated variant of the conditional generative adversarial network (CGAN), named pix2pixHD, which is capable of handling large-scale image resolutions that are critical for large FPGA layouts. This approach allows for accurate congestion predictions without the need for extensive feature engineering or intermediate routing stages, offering a direct visual assessment tool for designers to optimize layouts preemptively.

One of the main benefits of this technique is its potential to integrate directly into existing placement engines, providing real-time feedback and significantly reducing the number of iterations required to finalize the design. This can lead to improvements in routed wirelength by up to 7%, demonstrating substantial practical benefits in terms of efficiency and resource utilization in FPGA design. However the computational resources of this approach needed to handle large-scale image translations which could limit its application in less equipped settings.

To optimize routability with in global placement, the Reference [36] introduces a novel approach to enhancing the routability during the global placement phase of EDA design using deep learning techniques. The paper details the development and application of a fully convolutional network (FCN) designed to predict routing congestion from placement data and integrate these predictions into the placement process.

This method explicitly incorporates a routability model into the placement engine, termed DREAMPlace, which allows for adjustments in placement to minimize potential routing congestion. By predicting congestion hotspots and adjusting placements preemptively, the method achieves significant reductions in congestion rates and improves the routed wire length, showcasing up to 9.05% reduction in congestion and 5.30% improvement in wire length.

The deep learning model operates by using input features from the current placement to predict congestion, which is then used to guide the placement engine to adjust layouts to avoid potential congestion areas. This integration demonstrates an effective use of machine learning to directly influence and improve the VLSI design process, making it a practical tool for modern electronic design automation (EDA).

Despite the effectiveness of above 4 studies, there's the flaw that all of them using the crafted features (pin density map/net density map/RUDY map), which give a strong advantages to some models but extremely easy to lost netlist information for some models. In the meanwhile, the limitation of CNN itself that it only focusing on geometrical space and topological connection lost completely would cause several constraints to the experiment results. Nevertheless, the last Reference [37] presents the model named LHNN (Lattice Hypergraph Neural Network) leverages the combined strengths of lattice graphs and hypergraphs as shows in Figure 4.16 to preserve and utilize netlist data throughout the learning process, first net density maps are generated and combined to a 3-net circuit example, later hypergraph and lattice graph part of the circuit are transformed and integrated to LH-graph and the correlated schema finally, this procedure effectively capturing both geometric and topological relationships within the circuit layout.

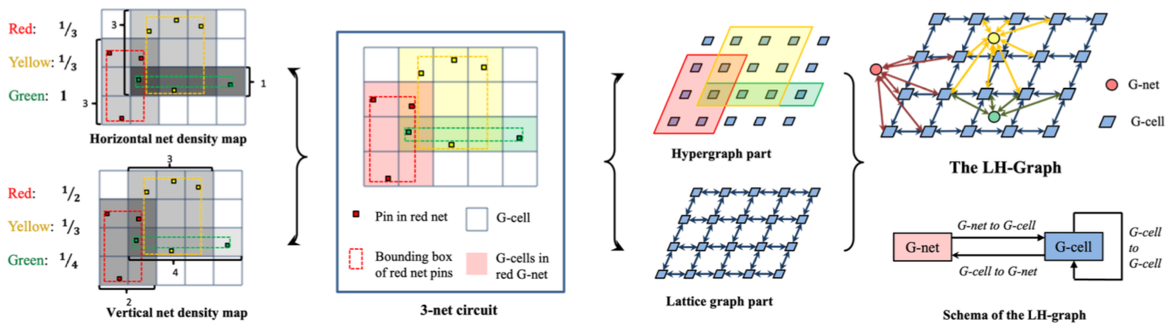

Figure 4.16: LH-Graph formulation and connection with crafted features

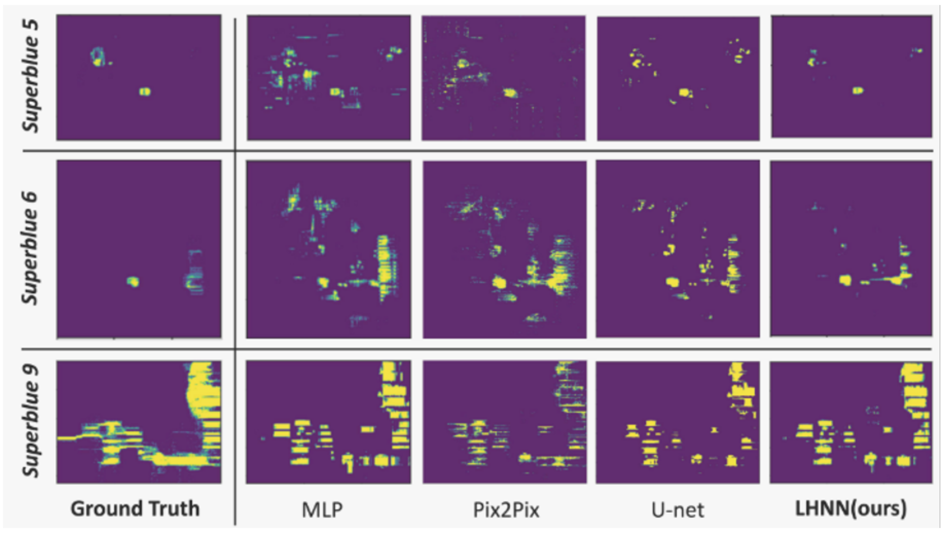

Figure 4.17: The comparison between Ground Truth with best performed model LHNN and other fundamental models

The LHNN model is designed to address several limitations inherent in traditional CNN-based models by facilitating the propagation of congestion information through a network that integrates both spatial and relational data. The model operates on a heterogeneous graph structure that enhances the receptive field beyond mere geometric adjacency, allowing for a deeper understanding of the circuit's inherent connectivity and potential congestion points.

A key advantage of this approach is its ability to predict congestion more accurately by maintaining the complete netlist information during training, which prevents the loss of critical data through feature conversion processes typically used in other models. The LHNN significantly outperforms traditional methods, achieving over 35% improvement on the F1 score compared to CNN models on benchmark datasets. This improvement highlights the model's efficacy in integrating complex relational data for congestion prediction. At last, the comparison between the fundamental models as MLP, Pix2Pix and U-net as mentioned in the previous papers with the Ground Truth and LHNN which proposed in the last paper is showed in the Figure 4.17, from which we can obviously obtain that LHNN outcomes other models completely and most similar to the ground truth.

The studies on machine learning applications in the routing phase highlight advancements in predicting and mitigating issues like congestion and DRC violations using sophisticated models such as deep learning, graph neural networks, and generative adversarial networks. These approaches allow designers to identify potential problems early, enhancing the routability and efficiency of circuit layouts. While these methods improve prediction accuracy and reduce design iterations, they also face challenges like high computational demands and the need for extensive training data. Despite these challenges, integrating machine learning into routing processes promises significant enhancements in reducing design cycles and improving manufacturability in electronic design automation.

CHAPTER 5

Conclusion

The use of ML methods for CAD has become an active area of research. Due to the ever increasing complexity and scale of variables in an IC design process, there is an increasing need for efficient ML/AI-assisted EDA tools. Moreover, with the event of emerging hardware security threats, there is a growing need for EDA tools to incorporate security countermeasures and mitigation techniques into the IC design flow. Although there has been significant progress in the development of tools and methods for hardware security, the need for efficient, easy to integrate, and scalable EDA tools is growing. With growing threats like IC counterfeiting, overproduction, reverse engineering, hardware trojan insertion, and side-channel attacks, the need to implement security mitigations and countermeasures into the IC design process is one to be addressed. In summary, this paper provides an insight on the advancements of using ML algorithms for EDA. The survey is categorized by the different IC design flow stages separately in Functional Simulation, Formal Verification, Logic Synthesis, Placement, Routing. Finally, Each section of the review has detailed how ML is being applied to solve specific challenges inherent in each step, demonstrating a clear trend towards the incorporation of advanced data-driven methodologies to enhance the efficiency, accuracy, and reliability of electronic design automation (EDA).

Functional simulation has benefited from ML in predicting circuit behavior more accurately and efficiently, while formal verification has seen improvements in automating and optimizing verification processes to ensure design correctness without exhaustive manual checks. Logic synthesis has been redefined by ML models that optimize circuit layouts and configurations to improve performance and reduce power consumption. In placement, ML techniques help predict and optimize the layout of components to minimize delays and improve manufacturability. Finally, in routing, ML applications focus on predicting routing congestion and DRC hotspots, enabling preemptive adjustments that streamline the routing process and reduce the need for costly post-routing corrections.

Across all these steps, the use of various ML approaches, including deep learning, reinforcement learning, and graph neural networks, has shown significant promise in transforming traditional EDA processes. These methods not only automate many of the tedious and complex tasks involved in VLSI design but also offer the potential to uncover solutions that may not be intuitive to human designers.

However, the review also highlights the challenges that come with the adoption of ML in EDA, such as the need for substantial computational resources, the dependence on large and high-quality datasets for training, and the complexities involved in integrating these models into existing workflows. Despite these challenges, the ongoing advancements in ML present a substantial opportunity to further enhance the capabilities and efficiency of EDA tools.

Overall, the incorporation of machine learning into EDA represents a significant shift towards more automated, intelligent, and efficient design processes, holding the promise of keeping pace with the increasing complexity of modern semiconductor devices and systems. As this field continues to evolve, further research and development in integrating ML with EDA will undoubtedly continue to reshape the landscape of electronic design.

Bibliography

[1] Rongjian Liang, Nathaniel Pinckney, Yuji Chai, Haoxin Ren, and Brucek Khailany. Late breaking results: Test selection for rtl coverage by unsupervised learning from fast functional simulation. pages 1–2, 07 2023.

[2] Blaise-Pascal Tine, Sudhakar Yalamanchili, and Hyesoon Kim. Tango: An optimizing compiler for just-in-time rtl simulation. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 157–162, 2020.

[3] Fubing Mao, Yapu Guo, Xiaofei Liao, Hai Jin, Wei Zhang, Haikun Liu, Long Zheng, Xu Liu, Zihan Jiang, and Xiaohua Zheng. Accelerating loop-oriented rtl simulation with code instrumentation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12):4985–4998, 2023.

[4] Wenjie Zhang, Zeyu Sun, Qihao Zhu, Ge Li, Shaowei Cai, Yingfei Xiong, and Lu Zhang. Nlocalsat: boosting local search with solution prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021.

[5] Emils Ozolins, Karlis Freivalds, Andis Draguns, Eliza Gaile, Ronalds Zakovskis, and Sergejs Kozlovics. Goal-aware neural sat solver. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[6] Vitaly Kurin, Saad Godil, Shimon Whiteson, and Bryan Catanzaro. Can q-learning with graph networks learn a generalizable branching heuristic for a sat solver? In *NeurIPS 2020: Proceedings of the Thirty-fourth Annual Conference on Neural Information Processing Systems*, December 2020.

[7] Emre Yolcu and Barnabás Póczos. Learning local search heuristics for boolean satisfiability. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] Eman M Elmandouh and Amr G Wassal. Guiding formal verification orchestration using machine learning methods. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 23(5):1–33, 2018.

[9] Weijun Zhu, Huanmei Wu, and Miaolei Deng. Ltl model checking based on binary classification of machine learning. *IEEE access*, 7:135703–135719, 2019.

[10] Honghao Gao, Baobin Dai, Huaikou Miao, Xiaoxian Yang, Ramon J Duran Barroso, and Hussain Walayat. A novel gapg approach to automatic property generation for formal verification: The gan perspective. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(1):1–22, 2023.

[11] Fnu Aditi and Michael S Hsiao. Hybrid rule-based and machine learning system for assertion generation from natural language specifications. In *2022 IEEE 31st Asian Test Symposium (ATS)*, pages 126–131. IEEE, 2022.

[12] Eman El Mandouh and Amr G Wassal. Estimation of formal verification cost using regression machine learning. In *2016 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, pages 121–127. IEEE, 2016.

[13] Yanqing Zhang, Haoxing Ren, and Brucek Khailany. Grannite: Graph neural network inference for transferable power estimation. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[14] MB Rakesh, Pabitra Das, Anant Terkar, and Amit Acharyya. Graspe: Accurate post-synthesis power estimation from rtl using graph representation learning. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2023.

[15] Ghasem Pasandi, Mackenzie Peterson, Moises Herrera, Shahin Nazarian, and Massoud Pedram. Deep-powerx: A deep learning-based framework for low-power approximate logic synthesis. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 73–78, 2020.

[16] Cunxi Yu and Wang 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*, pages 55–60, 2020.

[17] Abdelrahman Hosny, Soheil Hashemi, Mohamed Shalan, and Sherief Reda. Drills: Deep reinforcement learning for logic synthesis. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 581–586. IEEE, 2020.

[18] Walter Lau Neto, Max Austin, Scott Temple, Luca Amaru, Xifan Tang, and Pierre-Emmanuel Gaillardon. Lsoracle: A logic synthesis framework driven by artificial intelligence. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE, 2019.

[19] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. Developing synthesis flows without human knowledge. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[20] Winston Haaswijk, Edo Collins, Benoit Seguin, Mathias Soeken, Frédéric Kaplan, Sabine Süsstrunk, and Giovanni De Micheli. Deep learning for logic optimization algorithms. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2018.

[21] Zhiyao Xie, Rongjian Liang, Xiaoqing Xu, Jiang Hu, Yixiao Duan, and Yiran Chen. Net2: A graph attention network method customized for pre-placement net length estimation. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 671–677, 2021.

[22] Anthony Agnesina, Kyungwook Chang, and Sung Kyu Lim. Vlsi placement parameter optimization using deep reinforcement learning. In *Proceedings of the 39th international conference on computer-aided design*, pages 1–9, 2020.

[23] Yi-Chen Lu, Sai Pentapati, and Sung Kyu Lim. Vlsi placement optimization using graph neural networks. In *Proceedings of the 34th Advances in Neural Information Processing Systems (NeurIPS) Workshop on ML for Systems, Virtual*, pages 6–12, 2020.

[24] Anna Goldie and Azalia Mirhoseini. Placement optimization with deep reinforcement learning. In *Proceedings of the 2020 International Symposium on Physical Design*, pages 3–7, 2020.

[25] Yu-Hung Huang, Zhiyao Xie, Guan-Qi Fang, Tao-Chun Yu, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. Routability-driven macro placement with embedded cnn-based prediction model. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 180–185. IEEE, 2019.

[26] Xu He, Zhiyong Fu, Yao Wang, Chang Liu, and Yang Guo. Accurate timing prediction at placement stage with look-ahead rc network. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1213–1218, 2022.

[27] Erick Carvajal Barboza, Nishchal Shukla, Yiran Chen, and Jiang Hu. Machine learning-based pre-routing timing prediction with reduced pessimism. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[28] Peng Cao, Guoqing He, and Tai Yang. Tf-predictor: Transformer-based prerouting path delay prediction framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(7):2227–2237, 2022.

[29] Dani Maarouf, Abeer Alhyari, Ziad Abuowaimer, Timothy Martin, Andrew Gunter, Gary Grewal, Shawki Areibi, and Anthony Vannelli. Machine-learning based congestion estimation for modern fpgas. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 427–4277. IEEE, 2018.

[30] Robert Kirby, Saad Godil, Rajarshi Roy, and Bryan Catanzaro. Congestionnet: Routing congestion prediction using deep graph neural networks. In *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 217–222. IEEE, 2019.

[31] Rongjian Liang, Hua Xiang, Diwesh Pandey, Lakshmi Reddy, Shyam Ramji, Gi-Joon Nam, and Jiang Hu. Drc hotspot prediction at sub-10nm process nodes using customized convolutional network. In *Proceedings of the 2020 International Symposium on Physical Design*, pages 135–142, 2020.

[32] Kyeonghyeon Baek, Hyunbum Park, Suwan Kim, Kyumyung Choi, and Taewhan Kim. Pin accessibility and routing congestion aware drc hotspot prediction using graph neural network and u-net. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

[33] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. Routenet: Routability prediction for mixed-size designs using convolutional neural network. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[34] Jingsong Chen, Jian Kuang, Guowei Zhao, Dennis J-H Huang, and Evangeline FY Young. Pros: A plug-in for routability optimization applied in the state-of-the-art commercial eda tool using deep learning. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–8, 2020.

[35] Mohamed Baker Alawieh, Wuxi Li, Yibo Lin, Love Singhal, Mahesh A Iyer, and David Z Pan. High-definition routing congestion prediction for large-scale fpgas. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 26–31. IEEE, 2020.

[36] Siting Liu, Qi Sun, Peiyu Liao, Yibo Lin, and Bei Yu. Global placement with deep learning-enabled explicit routability optimization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1821–1824. IEEE, 2021.

[37] Bowen Wang, Guibao Shen, Dong Li, Jianye Hao, Wulong Liu, Yu Huang, Hongzhong Wu, Yibo Lin, Guangyong Chen, and Pheng Ann Heng. Lhnn: Lattice hypergraph neural network for vlsi congestion prediction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1297–1302, 2022.

CHAPTER 6

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor LAVAGNO LUCIANO, whose guidance,support and respect for my interested areas were invaluable throughout the duration of this thesis. Your sight and expertise have profoundly shaped this work and my view of academic works in the future, in the meanwhile, my growth as a scholar. It's such an honor to me to complete this work under the your guidance sir.

I an immensely grateful to the tutor of my thesis, Mr.Muhammad Usman Jamal, his professional advice, logical academic mindset and his generous spirit and kindly personality has incredibly improved the quality of this thesis and maintained my mental health, thanks boss.

I also want to extend my thanks to my friends Connor and Sina, who have been providing stimulating and supportive suggestions to me, thank you guys for being such reliable friends all the time.

A heartfelt thank you to my family and my girlfriend Leda Liu for their unwavering support and understanding throughout my master-academic journey. To my parents, thank you for your endless encouragement and belief in me. To my girlfriend, Leda Liu, your love, patience and support have been my anchor during the most challenging times.

Time has passed as a no-returning river, within which I've dedicated lots of efforts and suffered to obtain knowledge to be a qualified EE, luckily, I've harvested such beautiful things with everyone, which would be the life-time presents to me, thank you all.