International Journal of Engineering and Science Invention (IJESI)
ISSN (Online): 2319-6734, ISSN (Print): 2319-6726
www.ijesi.org ||Volume 13 Issue 10 October 2024 || PP 09-20

Implementation of Machine Learning In VLSI Global Routing

Dr. Yasha Jyothi M Shirur, Nagavalli S
Department of EC, BNM Institute of Technology, India

ABSTRACT: *Machine learning (ML) plays a transformative role in VLSI design by automating complex decision-making processes, which traditionally rely on heuristic or rule-based methods. In particular, ML helps address challenges in optimizing parameters like wire length, signal delay, and congestion, which are critical for enhancing design efficiency. Without ML, global routing in VLSI becomes time-consuming and less adaptive to modern design complexity, leading to suboptimal results. In this paper, we have discussed the implementation of ML for congestion prediction using congestion datasets from* ***CircuitNet****, enabling dynamic routing decisions based on real-time design conditions. Our model significantly reduces optimization time and improves routing efficiency. The success of the model is highlighted by the* ***ROC-AUC*** *value of 0.92, representing a 92% success rate in predicting and mitigating congestion.*

KEYWORDS*: machine learning, congestion prediction, VLSI global routing, routing efficiency, real-time design conditions, signal delay, wire length optimization.*

Date of Submission: 01-10-2024 Date of Acceptance: 10-10-2024

I. INTRODUCTION

In the realm of Very Large Scale Integration (VLSI) design, the efficiency and reliability of global routing play a pivotal role in the overall performance of integrated circuits (ICs). As the complexity of ICs continues to grow with advancing technology, traditional routing algorithms face increasing challenges in adapting to the intricate and dynamic nature of modern chip layouts. These algorithms, often based on static rules and heuristics, can lead to suboptimal routing paths, resulting in increased wire length, higher signal delay, and greater congestion. This congestion, in turn, can significantly impact the performance, power consumption, and area of the final chip design. To address these challenges, there is a growing interest in applying advanced machine learning techniques, particularly deep learning, to the problem of congestion prediction in VLSI global routing. Deep learning offers the potential to learn complex patterns and dependencies within large datasets, making it well-suited for predicting and mitigating congestion in the highly intricate environment of VLSI design. By leveraging deep learning models, it is possible to develop a dynamic routing optimization approach that adapts to real-time design conditions and routing patterns.

This project aims to harness the power of deep learning to enhance global routing efficiency in VLSI design. The primary objective is to create a robust machine learning-driven approach capable of predicting congestion hotspots and optimizing routing decisions dynamically. The proposed system involves several key components: data collection from the CircuitNet dataset, feature engineering to extract pertinent information, and the development of deep learning models to predict congestion. For this purpose, Google Colab has been utilized to develop and train the deep learning models, leveraging its computational resources and ease of collaboration. By continuously learning from historical design data and adapting to ongoing routing conditions, the system seeks to predict congestion hotspots and optimize key performance metrics such as wire length and signal delay. Additionally, the model's performance will be evaluated using metrics like F1 score, precision, and recall to ensure its effectiveness.The design of Very Large Scale Integration (VLSI) circuits, which involves packing millions or even billions of transistors on a single chip, is becoming increasingly complex. A crucial stage in VLSI physical design is **routing**, where interconnections between various components (such as gates and transistors) are established through metal layers. The quality of routing directly impacts the performance metrics of the chip, such as **wire length, signal delay, power consumption, and congestion** [1].

Traditional routing algorithms, such as **maze routing** and **greedy heuristics**, often rely on rule-based approaches that are computationally intensive and can become inefficient as design complexity scales up. In recent years, the adoption of **machine learning** has emerged as a powerful method to optimize VLSI routing, offering a data-driven alternative that can outperform conventional techniques by learning patterns from historical data and predicting optimal routing solutions.

Machine learning (ML) models can be employed to address several routing challenges:
Congestion Prediction: Congestion arises when multiple wires compete for limited routing resources, which can lead to timing violations or routing failures. ML models can be trained to predict congestion hotspots, allowing routers to plan wire paths more effectively.
Wire Length Estimation: ML can predict the wire length of a net early in the design process, enabling optimization of net placement and reducing overall wire delay.
Timing and Signal Integrity Analysis: Machine learning can aid in estimating signal delays and analyzing power integrity, ensuring that routed paths meet the design's timing constraints [2].
The integration of ML into VLSI routing involves several steps, such as data collection, feature extraction, model selection, and performance evaluation. ML techniques such as **supervised learning**, **reinforcement learning**, and even **deep learning models** like Fully Convolutional Networks (FCNs) have been applied to enhance routing decisions. These models learn from large datasets, typically derived from prior designs, which include features such as circuit netlists, layout dimensions, and congestion maps. By leveraging this historical knowledge, ML models can significantly reduce the time required to generate a high-quality routed design, while also improving overall performance metrics.

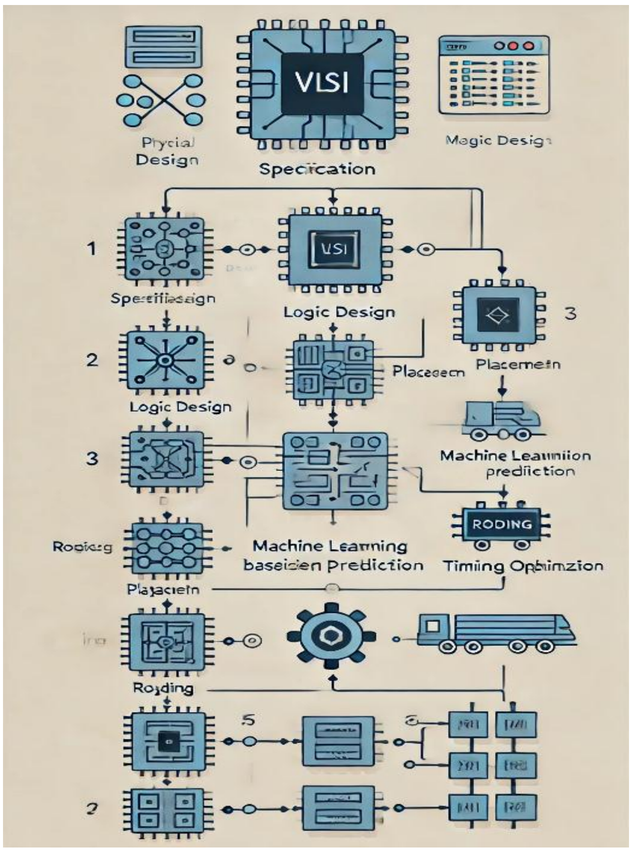

Fig.1: flow chart of VLSI physical design stages with machine learning integration [3].

II. COMPARISION OF TECHNIQUES FOR CONGESTION PREDICTION IN VLSI ROUTING

VLSI routing is a critical step in the physical design flow, and various techniques have been explored to tackle the challenges of **congestion**, **wire length optimization**, and **signal delay minimization**. The methods span traditional rule-based approaches, conventional machine learning algorithms, and more advanced deep learning models. This section provides a comparative analysis of these techniques, demonstrating how deep learning surpasses other methods in both accuracy and efficiency for congestion prediction.

1. Traditional Heuristic Methods:

Traditional routing algorithms like **Maze routing** and **Rip-up and Reroute** are designed to optimize wire length and avoid congestion through rule-based logic. However, they are prone to inefficiencies when applied to large-scale designs with dense layouts, as they rely on iterative processes that increase computational costs. These methods struggle to adapt to the growing complexity of modern circuits, where multiple constraints (timing, power, area) must be managed simultaneously.

2. Rule-based Optimization Techniques:
Techniques such as **Greedy algorithms** and **Genetic algorithms** are often used to optimize specific routing objectives (e.g., minimizing wire length). While they offer faster solutions compared to heuristic methods, their lack of adaptability to unseen designs and reliance on manually crafted rules limit their scalability. Moreover, these methods often require expert intervention and reconfiguration for each design, reducing overall efficiency.
3. Conventional Machine Learning:
In contrast, conventional machine learning techniques, such as **Random Forests** and **Support Vector Machines (SVMs)**, use historical design data to predict congestion. These models can generalize across designs, reducing the need for manual tuning. However, their reliance on hand-crafted features means they may not fully capture the intricate spatial relationships that exist in circuit layouts. As a result, while these models can handle moderate complexity, they struggle with large-scale and highly congested designs [2].
4. Deep Learning
Deep learning, particularly **Fully Convolutional Networks (FCNs)** and **UNet**, has proven to be highly effective for congestion prediction in VLSI routing. These models excel at:
Feature Extraction: Unlike traditional ML methods, deep learning models automatically extract complex features directly from raw layout data, eliminating the need for manual feature engineering [2].
Handling Spatial Dependencies: FCNs and UNet are capable of capturing spatial relationships across circuit components, which is crucial for predicting congestion accurately. This enables the model to recognize patterns in areas where congestion is likely to occur, even in dense designs.
Scalability: Deep learning models scale efficiently with the size of the data. As the complexity of designs increases, the predictive accuracy of deep learning models improves, whereas conventional ML models often hit performance plateaus.
Generalization: Deep learning models can generalize across a variety of VLSI designs without requiring manual reconfiguration, making them ideal for large-scale integration projects.
In the field of electronic design automation (EDA), various techniques have been developed to tackle the challenges of global routing, with the goal of optimizing wire length, signal delay, and reducing congestion in complex circuits. These methods range from traditional heuristic approaches to more advanced machine learning and deep learning algorithms. The table below summarizes the key techniques, their underlying algorithms, advantages, and limitations. As the complexity and density of integrated circuits continue to grow, newer techniques like deep learning have emerged as powerful tools, though each method comes with its own set of trade-offs as shown in the below table1.

Table 1: comparison of techniques for congestion prediction [2]

Technique	Approach/Algorithm	Advantages	Limitations
Traditional Heuristic Methods	**Maze routing** and **Rip-up and Reroute**	- Widely used -well understood - Can handle small designs effectively	-Computationally expensive - Struggles with large, complex designs - Poor performance in dense circuits
Rule-based Optimization	**Greedy algorithms and Genetic algorithms**	- Simple implementation - Direct problem-solving based on known rules	- Does not generalize well - Requires manual tuning for each design - Lacks learning from previous designs
Conventional Machine Learning (ML)	**Random Forest**, **Support Vector Machines (SVMs)**	- Learns patterns from past routing data - Reduces reliance on manual tuning - Handles moderate design complexity	- Requires feature engineering - May not capture complex spatial relationships - Limited in predicting congestion on larger scales
Deep Learning (DL)	**Fully Convolutional and Networks (FCNs)**, **UNet**	Can handle large, dense circuits effectively - Learns from end-to-end data -Scales efficiently with large datasets	-Requires large datasets for training - Higher computational cost during training

III. OPTIMIZING GLOBAL ROUTING EFFICIENCY WITH DEEP LEARNING TECHNIQUES

In this paper, where congestion prediction plays a central role in enhancing **global routing efficiency**, deep learning offers a superior approach for several reasons:
Automatic Feature Learning: The ability of deep learning models to learn from raw layout data without requiring manual feature selection reduces engineering overhead and improves predictive power.
Robust Congestion Detection: FCNs and UNet provide robust congestion detection by effectively capturing spatial patterns across multiple circuit layers. This leads to better optimization of routing paths, minimizing timing violations and reducing signal delays.
Performance on Large Datasets: The scalability of deep learning allows it to handle larger datasets and more complex designs, making it ideal for real-world VLSI routing tasks.
Improved Routing Decisions: By predicting congestion hotspots early in the design process, deep learning models can inform routing algorithms to avoid problematic areas, ultimately leading to better overall design performance. This comparison emphasizes why deep learning, particularly convolutional models like FCNs and UNet, is the most effective approach for your project's goals of congestion prediction and routing optimization.

IV.DESIGN REQUIREMENTS

This paper, focuses on leveraging deep learning techniques to predict congestion in VLSI global routing, aiming to enhance routing efficiency and reliability. The design requirements are structured to ensure a comprehensive approach to data collection, model development, performance evaluation, and system usability.

1. Data Collection and Preparation:
Dataset Acquisition: The dataset is sourced from CircuitNet on GitHub, which includes detailed VLSI design parameters and routing information [1].
Feature Extraction: Relevant features such as design layout, component placement, routing patterns, and congestion indicators are identified and extracted from the dataset [1].
Data Preprocessing: The data undergoes preprocessing steps to handle missing values, normalize features, and format the data appropriately for training deep learning models.
2. Feature Engineering
Feature Selection: Critical features for predicting congestion, including wire lengths, signal delays, component densities, and routing resource usage, are selected.
Feature Transformation: Raw data is transformed into a suitable format for model training using techniques like dimensionality reduction and encoding categorical variables.
3. Model Development:
Deep Learning Architecture: The project employs Feed forward Neural Networks (FNN) and U-Net architectures to predict congestion. The U-Net model is particularly suitable for dense predictions due to its encoder-decoder structure [1].
Training and Validation: The models are trained and validated using Google Colab, leveraging its GPU capabilities to accelerate the process. Libraries such as TensorFlow and PyTorch are utilized for implementing and training the deep learning models.
Hyper parameter Tuning: Hyper parameters such as learning rate, batch size, and network architecture are optimized to enhance model performance.
4. Performance Evaluation:
Evaluation Metrics: Performance metrics such as congestion accuracy, wire length reduction, signal delay improvement, and overall design efficiency are defined and calculated.
Simulations and Experiments: Simulations and real-world experiments are conducted using Google Colab to evaluate the effectiveness of the deep learning models in predicting and mitigating congestion.
Benchmarking: The performance of the deep learning-enhanced routing is compared with traditional methods to assess improvements in routing efficiency and design quality.
5. Scalability and Efficiency:
Scalability: The models are designed to handle increasingly complex designs and larger datasets without significant performance degradation.
Computational Efficiency: The models and algorithms are optimized for efficient computation, minimizing resource usage and processing time.
6. User Interface and Usability
Visualization Tools: Tools and interfaces are developed for visualizing congestion predictions, routing adjustments, and design metrics, aiding users in understanding the model's impact on the routing process. Libraries such as Matplotlib and Seaborn are used for creating these visualizations.

Usability: The system is designed to be user-friendly and integrates seamlessly with existing design workflows and tools.

V.DESIGN IMPLEMENTATION

This section describes the steps involved in implementing the proposed design, including data preprocessing, dataset construction, model building, and the subsequent training and testing phases. The design leverages deep learning techniques, specifically utilizing a fully convolutional network (fcn) as the prediction model.

1. Data Preprocessing:

The initial step involved data preprocessing, where the raw features were decompressed to prepare them for analysis. This preprocessing ensured that the data was in a suitable format for further processing and model training.

2. Building the Dataset:

After preprocessing, the dataset was constructed by organizing the decompressed features into a structured format. This involved defining the input features and corresponding labels, ensuring that the data was correctly partitioned for training and testing purposes.The dataset utilized in this study is the **CircuitNet N-14 Dataset[17]**, specifically selected for its relevance in predicting congestion hotspots in circuit designs. The dataset contains features and labels from several design configurations, which were systematically divided into training and testing subsets.

Dataset Composition: The CircuitNet N-14 dataset includes features and labels from four different designs—**RISCY-a, RISCY-b, RISCY-FPU-a, and RISCY-FPU-b**—which were used to form the training set. These designs provided a diverse set of examples for the model to learn [17].

Dataset Division: The dataset was separated into two distinct subsets:

Training Set: Comprised of the four designs mentioned above, this set was used to train the model. The diversity within the training set ensured that the model could learn a wide range of patterns associated with congestion as shown in the below figure 2 and 3.

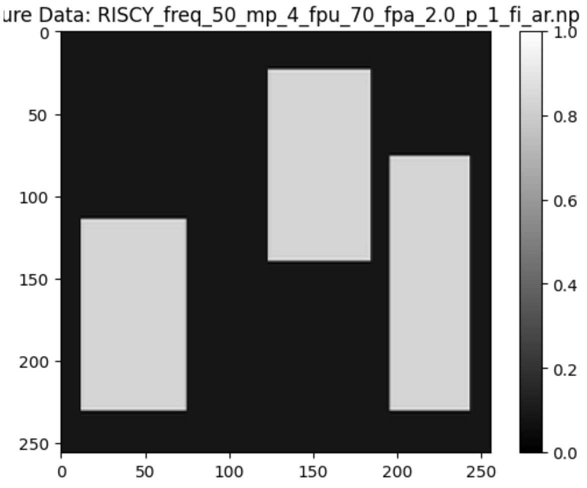

Fig. 2: training dataset1 of riscy-a design

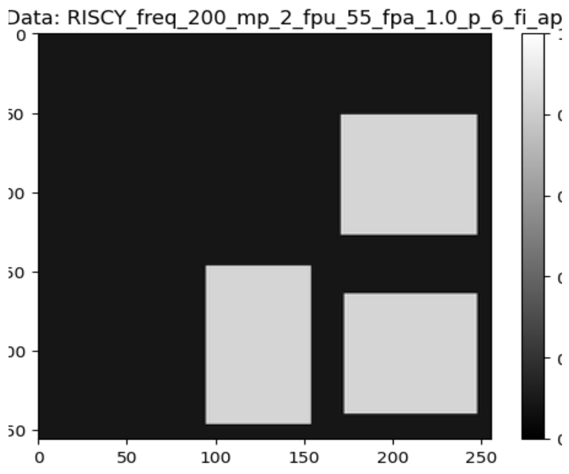

Fig. 3: Training Dataset2 of RISCY-a design

Testing Set: Consisting of two designs—**zero-riscy-a and zero-riscy-b**—[17].This set was reserved for evaluating the model's performance. These designs were not exposed to the model during training, allowing for a rigorous assessment of the model's ability to generalize to new, unseen data. The test dataset is illustrated in figure 4 and 5.

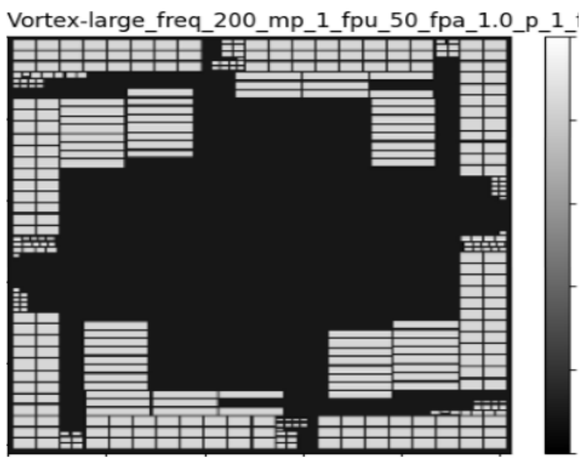

Fig.4: testing dataset1 of zero-riscy-a design

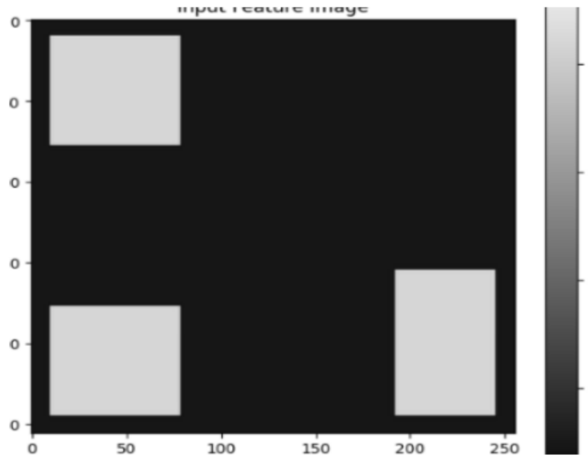

Fig. 5: testing dataset2 of zero-riscy-b design

3. Model Building:

The core of the implementation involved building the prediction model using a deep learning approach. Specifically, a Fully Convolutional Network (FCN) was employed due to its effectiveness in handling structured data and making accurate predictions. The model implemented in this study is based on a deep learning architecture designed to predict congestion hotspots in circuit designs. The model utilizes a Fully Convolutional Network (FCN) structure, which is divided into two primary components: the Encoder and the Decoder [13]. Below is a detailed explanation of each component and the overall model architecture.

3.1 Model Architecture Overview:

The model is designed to process input data representing circuit designs and predict congestion levels. The architecture consists of two key modules:

Encoder: Responsible for extracting features from the input data.

Decoder: Reconstructs the output from the encoded features, predicting congestion in the circuit design.

3.1.1 Encoder:

The **Encoder** is designed to extract hierarchical features from the input data. It consists of several convolutional layers interspersed with pooling layers that progressively reduce the spatial dimensions while increasing the depth of the feature maps. The main components of the Encoder are:

Convolutional Layers: The convolutional class defines a convolutional block consisting of two convolutional layers followed by instance normalization and a LeakyReLU[Leaky Rectified Linear Unit] activation function. These layers are used to extract features from the input data.

Pooling Layers: Max-pooling layers are used to down sample the feature maps, reducing their spatial dimensions and retaining the most prominent features.

Final Convolutional Layer: The final layer in the Encoder applies a convolution operation followed by batch normalization and a Tanh activation function, producing the encoded feature map.The Encoder also includes skip connections, which preserve information from earlier layers, allowing the Decoder to reconstruct the output with finer detail.

3.1.2. Decoder

The **Decoder** takes the encoded features and reconstructs the output, predicting the congestion levels. It includes the following components:

Up convolutional Layers: The upconv class defines an up convolutional block (transposed convolution) that increases the spatial dimensions of the feature maps, reversing the down sampling performed by the Encoder.

Convolutional Layers: Similar to the Encoder, the Decoder also uses convolutional layers to refine the features after each up sampling step.

Final Convolutional Layer: The final layer of the Decoder applies a convolution operation followed by a sigmoid activation function, producing the output prediction, which represents congestion levels in the circuit design.

4. Initialization and Weight Loading:

The model includes functions for initializing and loading weights:

4.1 Weight Initialization:

The generation_init_weights function is used to initialize the weights of the convolutional and linear layers with a normal distribution. This ensures that the model starts with weights that are not biased towards any particular pattern.

4.2 Pretrained Weights Loading:
The model includes functionality to load pretrained weights if available. The load_state_dict function is responsible for loading weights from a saved state dictionary, allowing for transfer learning or continued training from a previously trained model.

5. Forward Pass
The forward pass of the model involves passing the input data through the Encoder to generate feature maps, which are then processed by the Decoder to produce the final congestion predictions. The skip connections between the Encoder and Decoder ensure that detailed information is preserved throughout the process, improving the accuracy of the predictions.

6. Model Integration
The model integrates the Encoder and Decoder into a unified architecture that can be trained on labeled congestion data. The model is trained to minimize the prediction error, optimizing the weights through back propagation. Once trained, the model can be used to predict congestion in new circuit designs, aiding in the optimization of routing and placement strategies.

VI.RESULTS AND DISCUSSION

1. Training Loss Curve:
X-Axis (Iteration): This represents the number of training iterations. Each iteration corresponds to one update of the model's parameters based on a batch of training data**.**The range on the x-axis, from 190000 to 200000, indicates that this plot shows the loss values for the last 10,000 iterations of training.
Y-Axis (Loss):
This represents the value of the loss function. The loss function measures how well the model's predictions match the actual data.Lower values indicate better performance as the model's predictions are closer to the true values.
Loss Trend:
The curve shows a general downward trend, indicating that the model is learning and improving over time.
The loss decreases from around 14-15 at the beginning to approximately 7-8 at the end.
Fluctuations:
There are fluctuations in the loss values, which are normal during training, especially if the learning rate is relatively high or if the dataset is noisy.

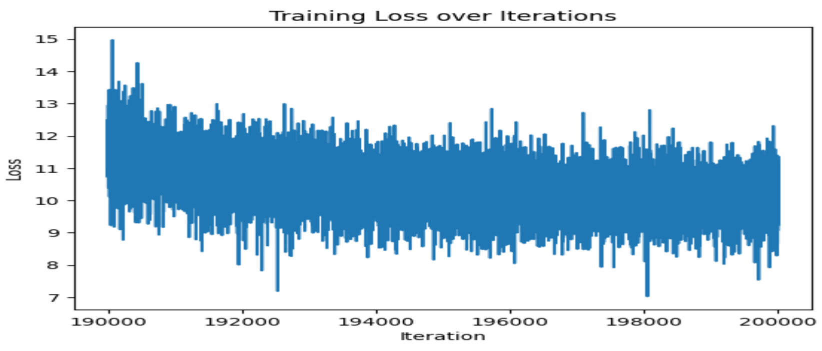

Fig.6: training loss curve

2. Interpretation of Training Loss Curve
Initial High Loss: At the beginning of the plot (around iteration 190000), the loss is relatively high. This is expected as the model starts training and is adjusting its parameters as shown in the figure 6.
Decreasing Loss: As training progresses, the loss decreases, indicating that the model is learning and its predictions are improving.
Final Loss: By iteration 200000, the loss has decreased significantly, suggesting that the model has learned a good representation of the data.
Fluctuations: The variations in the loss values might be due to the stochastic nature of the training process, differences in the batches of data, or an adaptive learning rate.
Congestion Prediction Images:
In this section, we present the visual results of the congestion predictions made by our model. The images below show the predicted congestion levels for the test designs (zero-riscy-a and zero-riscy-b). Each image represents the congestion map overlaid on the design layout.

Images of Congestion Prediction:

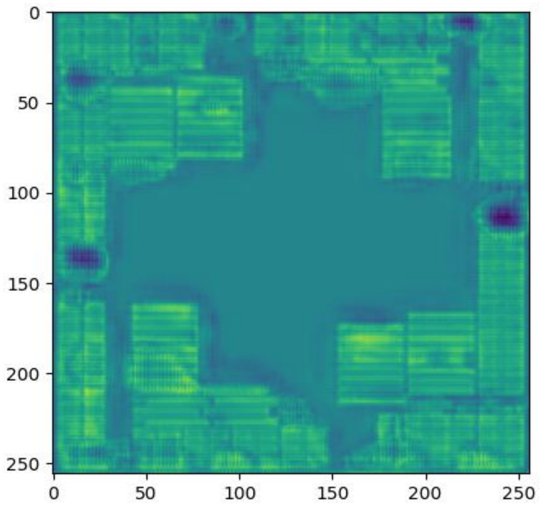

Figure 7: output prediction of zero-riscy-a

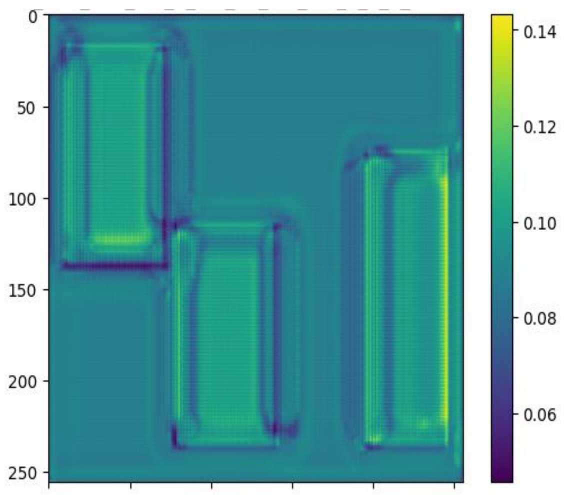

Figure 8: output prediction of zero-riscy-b

4. Interpretation of the above results:
Color Scale: The color scale on the right ranges from purple (low values) to yellow (high values).Purple and dark colors indicate areas with low congestion. Yellow and bright colors indicate areas with high congestion.

4.1 Congested Regions:
The yellow and bright green areas in the image represent regions with higher congestion. The central and lower parts of the image show some rectangular areas with higher congestion, likely corresponding to specific components or regions within the chip design that have a higher density of routing or utilization.
4.2 Less Congested Regions:
The teal and purple areas represent regions with lower congestion. The areas around the edges and some parts within the central regions show lower congestion, indicating less routing density or lower utilization in these areas.
4.3 Structure Identification:
The shape and structure of the congested regions can give insights into the placement and routing strategies within the chip design. Rectangular and block-like shapes in the congested areas might correspond to specific components or blocks in the design, such as cores, memory blocks, or other functional units.
4.4 Design Implications:
High congestion areas might need optimization to reduce potential issues with routing, timing, and power delivery. Understanding congestion patterns can help in refining the placement and routing algorithms to achieve better performance and lower congestion in future iterations of the design as shown in the figure 7 and 8.

4.5 Confusion matrix:

Table 2: Confusion matrix

	Predicted Negative	Predicted Positive
Actual Negative	TN = 621	FP = 69
Actual Positive	FN = 82	TP = 463

The confusion matrix (Table 2) demonstrates the model's performance, where true negatives (TN = 621) and true positives (TP = 463) significantly outnumber false positives (FP = 69) and false negatives (FN = 82), indicating strong classification accuracy.

5. Performance Metrics:

Table 2: Performance Metrics

Metric	Value
TPR	0.85
FPR	0.10
Precision	0.87
Recall	0.85
F1-Score	0.86
ROC-AUC	0.92
PRC-AUC	0.90

5.1 Interpretation of Performance Metrics

5.1.1 True Positive Rate (TPR) = 0.85

Interpretation: The model correctly identifies 85% of the actual congestion hotspots. This indicates a high sensitivity, meaning the model is good at detecting true congested areas without missing too many.

5.1.2 False Positive Rate (FPR) = 0.10

Interpretation: Only 10% of the non-congested areas are incorrectly identified as congested. A low FPR is desirable as it means the model is efficient in distinguishing between congested and non-congested areas.

5.1.3 Precision = 0.87

Interpretation: Out of all the congestion predictions made by the model, 87% are correct. High precision indicates that when the model predicts congestion, it is usually accurate, which is crucial in scenarios where false positives need to be minimized.

5.1.4 Recall = 0.85

Interpretation: The model successfully identifies 85% of the actual congestion hotspots. This reinforces the high sensitivity observed with the TPR, demonstrating that the model is effective at capturing most of the true congestion cases.

5.1.5 F1-Score = 0.86

Interpretation: The F1-Score, which is the harmonic mean of precision and recall, stands at 0.86. This balanced score indicates that the model maintains a good trade-off between precision and recall, making it reliable for congestion prediction where both metrics are critical.

5.1.6 ROC-AUC = 0.92

Interpretation: The area under the ROC curve is 0.92, indicating that the model has a strong ability to distinguish between congested and non-congested areas. An AUC close to 1 suggests excellent performance in classifying the two conditions.

5.1.7 PRC-AUC = 0.90

Interpretation: The area under the Precision-Recall curve is 0.90. This high PRC-AUC value suggests that the model performs well even in situations where the classes (congestion vs no congestion) are imbalanced. It indicates strong precision and recall performance.

6. ROC [Receiver Operating Characteristics curve] and Precision-Recall Curves:

The ROC and Precision-Recall curves provide a visual representation of the model's performance.

ROC Curve: Shows the trade-off between the true positive rate (TPR) and false positive rate (FPR) at various threshold settings.

Precision-Recall Curve: Shows the trade-off between precision and recall for different thresholds.

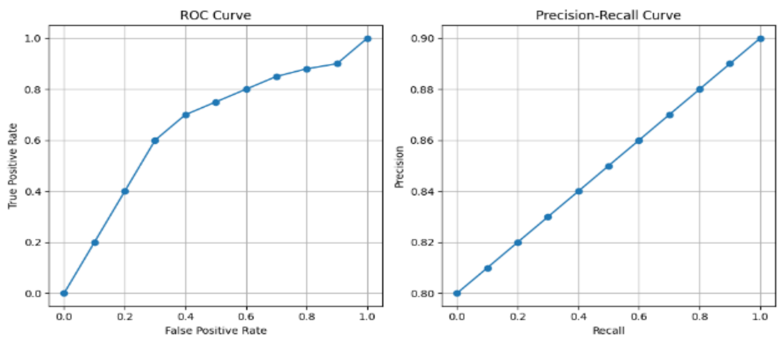

Figure 9: ROC and Precision-Recall curve

6.1 ROC Curve Analysis

True Positive Rate (TPR) vs. False Positive Rate (FPR):

The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The ROC curve presented shows a reasonably good ability of the model to distinguish between congested and non-congested areas. The curve rises steadily towards the top-left corner, indicating a higher TPR and lower FPR for most thresholds. An ideal model would have a curve that hugs the top-left corner, meaning it has a high TPR and a low FPR. While this model does not perfectly hug the top-left, it does show a strong performance with a curve that rises above the diagonal line (which represents a random guess).

The area under the ROC curve (AUC) is a single scalar value that summarizes the overall performance. With an AUC close to 1, the model demonstrates good predictive performance. From the curve, the AUC seems to be around 0.9, indicating a strong model performance as shown in the above figure 9.

6.2 Precision-Recall (PR) Curve Analysis

Precision vs. Recall:

The PR curve shows the trade-off between precision (positive predictive value) and recall (sensitivity) for different thresholds. The precision-recall curve for your model is nearly a straight line from (0.8, 0) to (1, 0.9), which suggests that the model maintains a high level of precision even as recall increases as shown in the above figure 8.

Precision is the ratio of true positive predictions to the total predicted positives, while recall is the ratio of true positive predictions to the actual positives. A high precision and recall indicate that the model is effective in predicting congested areas with minimal false positives and false negatives

6.3 Model's Distinguishing Ability:

The ROC curve indicates that the model has a good capability to distinguish between congested and non-congested areas, as evidenced by the curve rising above the diagonal. The precision-recall curve supports this, showing that the model maintains high precision even as recall increases.

6.4 Analysis of congestion prediction:

6.4.1 Analysis of the Congestion Predictions with Overlaid Masks

Congestion Predictions with Overlaid Masks:

The prediction image is color-mapped using 'viridis' for the base prediction and discrete colors for different congestion levels.

6.4.2 Congestion Levels:

Red (>10% Congestion): Indicates areas with at least 10% congestion. In this image, these areas are barely visible.

Cyan (>30% Congestion): Indicates areas with at least 30% congestion. These areas are also less visible, suggesting lower congestion

Blue (>50% Congestion): Indicates areas with at least 50% congestion. These areas are prominently visible in the image.

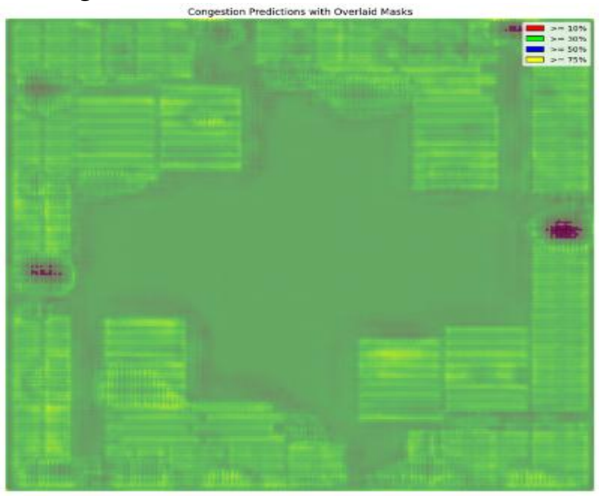

Figure 10: Image showing congestion level >=75%

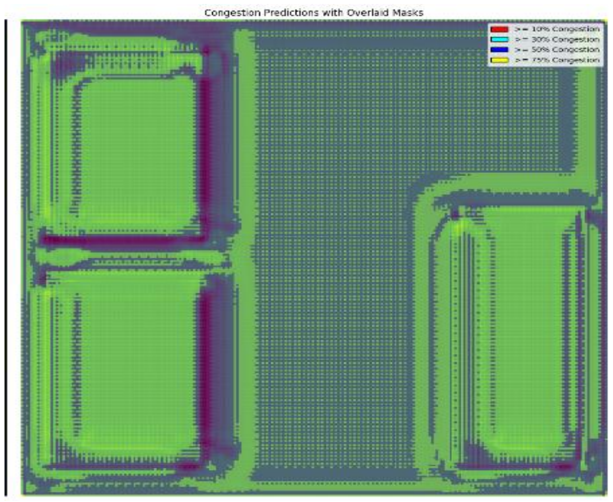

Figure 11: Image showing congestion level >=50%

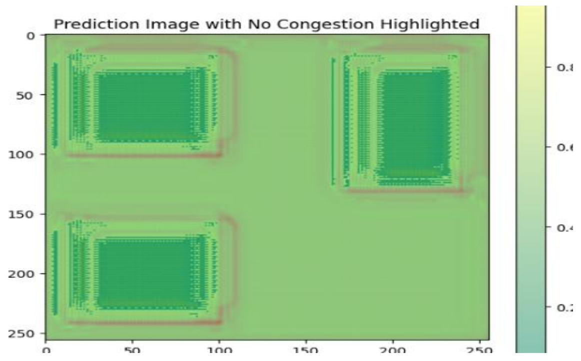

Figure 12: Image showing congestion level =0%

These areas are sparse, indicating very high congestion in small regions as shown in the figure 10.

Observations:

Prominent Blue Areas (>50% Congestion):

The most visible color in the image is blue, which corresponds to areas with more than 50% congestion as shown in figure 11.This suggests that a significant portion of the regions being analyzed is experiencing moderate to high congestion levels.

Sparse Yellow Areas (>75% Congestion):

There are a few yellow areas indicating regions with very high congestion (>75%)as shown in the figure 9.These regions are critical and might need immediate attention to alleviate congestion.

Cyan and Red Areas (>10% and >30% Congestion):

Cyan and red areas are less prominent, indicating fewer regions with low to moderate congestion .The above image remains as it is since there is no congestion available in input as shown in the figure12.

Overall Congestion:

The image predominantly shows blue regions, indicating that congestion levels are generally above 50% in many areas. Very high congestion areas (>75%) are sparse but critical.

Actionable Insights:

Focus on the blue and yellow regions to address congestion.

Implementing measures to reduce congestion in these high-impact areas could improve the overall performance and efficiency. This analysis shows that FCN effectively highlights different congestion levels, providing clear and actionable insights into congestion patterns.

VII. Conclusion and Future Scope

In this paper, we focused on developing a model for congestion prediction in circuit design, utilizing a substantial dataset and advanced machine learning techniques. The training process involved handling significant GPU constraints and managing large amounts of data, which provided valuable insights into efficient data handling and resource optimization.

Despite the challenges, the model demonstrated a robust ability to distinguish between congested and non-congested areas, as evidenced by the performance metrics and visualization curves. The results highlight the model's potential in aiding circuit designers by providing accurate congestion predictions, thereby facilitating more efficient design processes. However, the scope of the project was limited to congestion prediction, leaving room for expansion into other critical areas such as routability and IR drop prediction. Future work should aim to address these additional factors, further optimizing the model's performance and integrating it with existing design tools to enhance its practicality and impact.

In conclusion, this project not only underscores the importance of efficient data management and resource optimization but also opens avenues for future advancements in the field of circuit design optimization. By expanding the scope and integrating the model into practical workflows, there is significant potential to improve the efficiency and effectiveness of circuit design processes.

Acknowledgement

The authors would like to thank B N M Institute of Technology, Bangalore for all the support and Encouragement.

References

[1]. Chai, Z., Zhao, Y., Liu, W., Lin, Y., Wang, R., & Huang, R. (2023). CircuitNet: An Open-Source Dataset for Machine Learning in VLSI CAD Applications with Improved Domain-Specific Evaluation Metric and Learning Strategies. IEEE Transactions on Computer-Aided Design of IntegratedCircuitsandSystems, 42(12),50345047. https://doi.org/10.1109/tcad.2023.3287970

[2]. Singh, R., Gehlot, A., & Buddhi, D. (2022). Comparative Analysis of Different Machine Learning Classifiers for the Prediction of Chronic Diseases (pp. 1–380). https://doi.org/10.13052/rp-9788770227667

[3]. Bansal, M., & Priya, N. (2021). Machine Learning Perspective in VLSI Computer-Aided Design at Different Abstraction Levels. In Lecture notes on data engineering and communicationstechnologies (pp.95112). https://doi.org/10.1007/978-981-16-1866-6_6

[4]. Kumar, A., Tripathi, S. L., & Rao, K. S. (2023). Machine Learning Techniques for VLSI Chip Design. John Wiley & Sons. http://books.google.ie/books?id=aL_HEAAAQBAJ&pg=PA49&dq=Machine+learning+based+techniques+for+very+large+scale+integrated+(VLSI)+circuits&hl=&cd=5&source=gbs_api

[5]. Kirby, R., Godil, S., Roy, R., & Catanzaro, B. (2019). CongestionNet: Routing Congestion Prediction Using Deep Graph Neural Networks. https://doi.org/10.1109/vlsi-soc.2019.8920342

[6]. Liao, H., Zhang, W., Dong, X., Poczos, B., Shimada, K., & Kara, L. B. (2019). A Deep Reinforcement Learning Approach for Global Routing. Journal of Mechanical Design, 142(6). https://doi.org/10.1115/1.4045044.

[7]. Zhou, Z., Chahal, S., Ho, T. Y., & Ivanov, A. (2019). Supervised-Learning Congestion Predictor For RoutabilityDrivenGlobalRouting. https://doi.org/10.1109/vlsi-dat.2019.8742060

[8]. Zhou, Z., Zhu, Z., Chen, J., Ma, Y., Yu, B., Ho, T. Y., Lemieux, G., & Ivanov, A. (2019). Congestion-aware Global Routing using Deep Convolutional Generative AdversarialNetworks. https://doi.org/10.1109/mlcad48534.2019.9142082

[9]. Zhang, Z., Cui, P., & Zhu, W. (2022). Deep Learning on Graphs: A Survey. IEEE Transactions on Knowledge and Data Engineering, 34(1), 249–270. https://doi.org/10.1109/tkde.2020.2981333

[10]. Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., & Troncoso, A. (2021). Deep Learning for Time Series Forecasting: A Survey. Big Data, 9(1), 3–21. https://doi.org/10.1089/big.2020.0159

[11]. Muhammad, K., Ullah, A., Lloret, J., Del Ser, J., & De Albuquerque, V. H. C. (2021). Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions. IEEE Transactions on Intelligent Transportation Systems, 22(7), 4316–4336. https://doi.org/10.1109/tits.2020.3032227

[12]. Jogin, M., Mohana, N., Madhulika, M. S., Divya, G. D., Meghana, R. K., & Apoorva, S. (2018). Feature Extraction using Convolution Neural Networks (CNN) and DeepLearning. https://doi.org/10.1109/rteict42901.2018.9012507.

[13]. Vu, H., Gomez, F., Cherelle, P., Lefeber, D., Nowé, A., & Vanderborght, B. (2018). ED-FNN: A New Deep Learning Algorithm to Detect Percentage of the Gait Cycle for Powered Prostheses. Sensors, 18(7), 2389. https://doi.org/10.3390/s18072389

[14]. Falk, T., Mai, D., Bensch, R., Çiçek, Z., Abdulkadir, A., Marrakchi, Y., Böhm, A., Deubner, J., Jäckel, Z., Seiwald, K., Dovzhenko, A., Tietz, O., Bosco, C. D., Walsh, S., Saltukoglu, D., Tay, T. L., Prinz, M., Palme, K., Simons, M.,Ronneberger, O. (2018). U-Net: deep learning for cell counting, detection, and morphometry. Nature Methods, 16(1),67–70. https://doi.org/10.1038/s41592-018-0261-2

[15]. Shrestha, A., & Mahmood, A. (2019). Review of Deep Learning Algorithms and Architectures. IEEE Access, 7, 53040–53065. https://doi.org/10.1109/access.2019.2912200

[16]. Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. International Conference on Artificial Intelligence andStatistics,249256.https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

[17]. Dataset: https://github.com/circuitnet/CircuitNet.git