AiEDA-2.0: An Open-source AI-Aided Design (AAD) Library for Design-to-Vector

Yihang Qiu[2,1*], Zengrong Huang[1], Weiguo Li[1], Xinhua Lai[2], Rui Wang[3], He Liu[4], Ping Zhou[1], Simin Tao[1], Junfeng Liu[1], Yifan Li[1], Xingquan Li[1,✉]

[1]Pengcheng Laboratory, Shenzhen, China; [2]University of Chinese Academy of Sciences, Beijing, China
[3]Shenzhen University, Shenzhen, China; [4]Peking University, Shenzhen, China
Email: *qiuyihang23@mails.ucas.ac.cn, ✉lixq01@pcl.ac.cn

***Abstract*—Although artificial intelligence (AI) has made significant progress in the electronic design automation (EDA) field, specialized infrastructure remains insufficient. In this paper, we analyze the necessary components for the integration of AI with EDA, propose a data decomposition from design to vector, and build an open-source AI-aided design (AAD) library. This library aims to transform chip data into vectors, train AI4EDA models, and integrate trained models into the chip design flow.**

***Index Terms*—AI-aided chip design, open-source library**

I. Introduction

In the EDA domain, the intricacies of tasks such as design space exploration, logic optimization, placement, and routing exemplify the inherent challenges faced. Traditional algorithms, when applied to these tasks, often get stuck in sub-optimal solutions, necessitating compromises in quality or incurring prohibitive computational costs. However, the emergence of AI has introduced opportunities in the field of EDA. Leveraging AI's advanced search capabilities and sophisticated learning algorithms, AI demonstrates its transformative potential by delivering more efficient and higher-quality solutions [1].

Although AI has made considerable progress in EDA, the field still lacks specialized research infrastructure [2]. The industry needs specialized AI models tailored for EDA tasks, accessible labeled chip datasets, and AI libraries or packages for IC/EDA design. Recently, the Python APIs in iEDA [3] and OpenROAD [4] wrap their underlying C++ API to generate data more quickly. NVIDIA's CircuitOps [5] has introduced solutions for efficient data handling and can leverage the Python API in OpenROAD as a feedback loop, further enhancing interaction with machine learning algorithms. Therefore, establishing and enhancing these infrastructures is an urgent issue that needs to be addressed in future AI/EDA research and applications.

II. Design-to-Vector

The digital chip design flow converts RTL Verilog into a GDS-II layout. However, for AI tasks, traditional design formats may be insufficient. We require vectorized data such as vectors, matrices, and tensors. To address this, we propose a new data transformation method for AI4EDA, "design-to-vector," which converts chip designs into vectors. "design-to-vector" consists of several stages, as illustrated in Fig. 1. Given chip data, we run the chip design flow using EDA tools that generate intermediate formats such as GTech/AIG, netlist, DEF, GDS, and SPEF, which are stored as basic data. Additional features or metrics are then derived from the design process and stored in formats like JSON, CSV and so on. To improve AI models' ability to process and

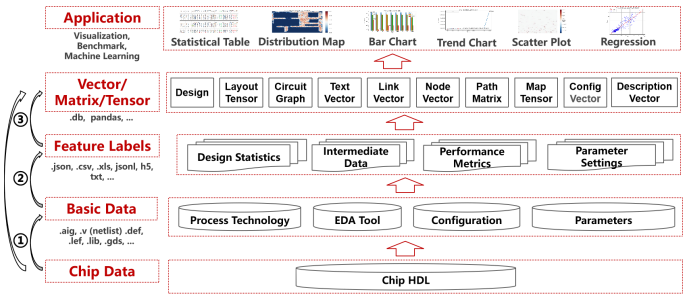


Fig. 1. Design-to-vector.

understand EDA tasks, it is essential to gather comprehensive data from the entire design process, including both results and features. These diverse features extracted from the chip design process are easily transferred to a vector format and saved into a vector database system, encompassing graphs, maps, vectors, matrices, and tensors. For the design results, we propose a vectorization technique to hierarchically decompose a design into wire sets. Fig. 2 illustrates the detailed process of vectoring a design: from netlist and layout to wire.

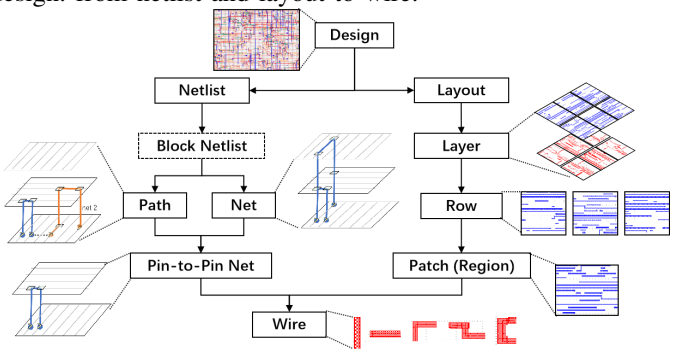


Fig. 2. Netlist/layout-to-wire.

III. AiEDA: AI-Aided Design (AAD) Library

Traditionally, computer-aided design (CAD) has significantly advanced chip design automation, and AI has now proven its strong capabilities in this area. To enhance AI-assisted design, we develop an open-source AI-aided design (AAD) library (AiEDA), which provides three key components: 1) Python APIs for EDA tools; 2) Vectorized data formats for effective representation and sharing; and 3) Evaluation of AI models within the chip design flow. The open-source repository of AiEDA is: https://github.com/OSCC-Project/AiEDA.

To develop AI models for EDA, developers are accustomed to using Python APIs provided by EDA tools. These APIs not

only enable easy access to data but also support efficient model training and feedback optimization, allowing AI researchers to engage more deeply with EDA tools and refine their models. Second, we need some easy-to-use tools or flows to convert chip data into vectorized data. Finally, a comprehensive evaluation system for trained AI models is essential. This evaluation system can easily integrate trained AI models into existing EDA tools and workflows. By comparing outputs, this system can assess performance and validate proposed solutions.

IV. AiEDA Applications

With AiEDA, we can easily extract various features, convert design data to vectors, build AI models for EDA tasks, and integrate these models into the chip design flow for evaluation and inference. We have successfully transformed 50 chip designs into wires, with the comprehensive statistics presented in Table I. Additionally, we can perform insightful data analysis and optimization of chip design, including data statistics, correlation analysis, similarity analysis, variable regression, design classification, metric fitting and prediction, design space exploration, and algorithm or tool optimization.

TABLE I
Design Statistics Summary

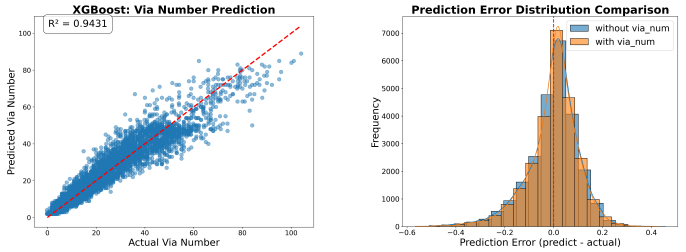
Designs	Cells	Nets	Pins	Wires	Nets		Patches		Paths	
					Num	Size	Num	Size	Num	Size
s38584	6K	7K	22K	98K	7K	61M	12K	90M	5K	63M
aes	17K	18K	67K	326K	18K	211M	36K	312M	12K	633M
eth_top	38K	39K	133K	647K	39K	433M	170K	910M	20K	562M
...45 more designs ...										
T1	1.3M	1.2M	4.2M	18M	1.2M	11G	2.4M	16G	736K	16G
C910	3.3M	2.9M	9.6M	52M	2.9M	33G	6.2M	53G	1.7M	30G
Total	23M	21M	72M	347M	21M	212G	52M	348G	12M	190G

Path Delay Prediction: From the path data, we can easily parse resistance, capacitance, and slew as features, with total path delay as labels. We employed Transformer and MLP as our base models, collecting 10,000 paths for training and 2,000 paths for testing. The final model achieved a mean absolute error of approximately 0.04, corresponding to a relative error of 7%.

Wirelength Prediction: From the net data, we can easily parse features from the placement stage, such as RSMT, pin number, and aspect ratio, with the ratio of post-routing wirelength to RSMT as labels. We used XGBoost as our base model, collecting 100,000 nets for training and 25,000 nets for testing. We also built a via prediction model with an R^2 value of about 0.94 (Fig. 3(a)). By adding the via prediction results as extra features to the wirelength model, we reduced the mean relative error by 4%, with the error distribution shifting to the left (Fig. 3(b)).

Design Rule Violation Prediction: Using the patch data, we can reconstruct full-layout features. Each patch serves as a pixel in the image, storing multiple features such as pin density and DRC violations. We used CNN as our base model to predict the violation distribution. The prediction results are shown in Fig. 4.

Routing Net Generation: The combination of net and patch data can enable various applications, as illustrated in Fig. 5(a). By representing the 3D net information in a 2D format and indexing the corresponding patches, we can access the patch-level features such as congestion. Based on this, we conducted a 2-pin routing generation task using a transformer-based model



(a) (b)

Fig. 3. Model performance evaluation: (a) via prediction model accuracy and (b) wirelength ratio error distribution comparison.

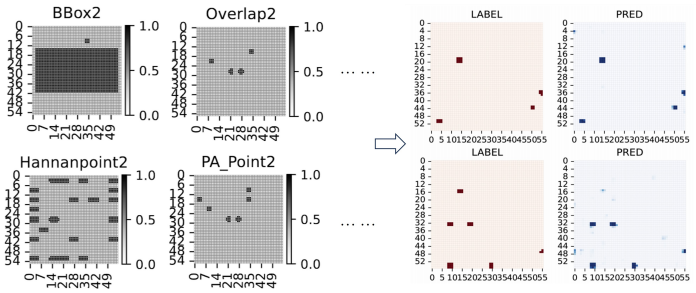
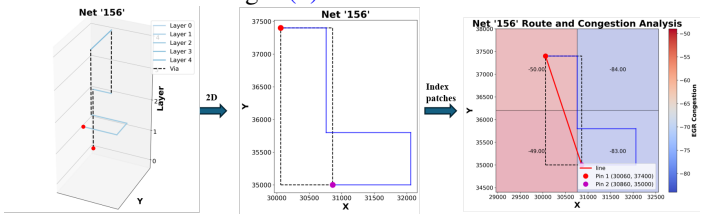
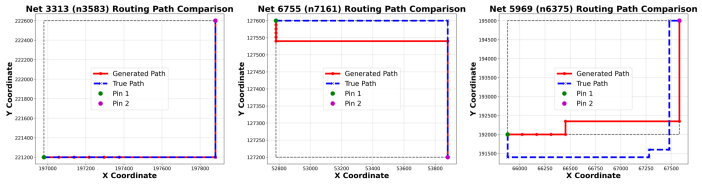


Fig. 4. Prediction of DRC violations using patch data.

that effectively learns the routing patterns by leveraging the combined net and patch features. The routing net generation results are shown in Fig. 5(b).



(a)



(b)

Fig. 5. Routing generation: (a) Integrating net and patch (b) Generation results.

V. Conclusions

We show a flow from design to vector. We build an open-source library (AiEDA) to achieve AI-aided design. We list some demos to show the possible applications of AiEDA.

References

[1] G. Huang, J. Hu, Y. He, et al. Machine Learning for Electronic Design Automation: A Survey. *ACM Trans. on DAES*, pp. 1–46, 2021.

[2] A. B. Kahng. Solvers, Engines, Tools and Flows: The Next Wave for AI/ML in Physical Design. *In Proc. of ISPD*, pp. 117–124, 2024.

[3] X. Li, Z. Huang, S. Tao, et al. iEDA: An Open-source Infrastructure of EDA. *In Proc. of ASP-DAC*, 2024.

[4] T. Ajayi, V. A. Chhabria, et al. Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project. *In Proc.of DAC*, pp. 1-4, 2019.

[5] R. Liang, A. Agnesina, et al. CircuitOps: An ML Infrastructure Enabling Generative AI for VLSI Circuit Optimization. *In Proc. of ICCAD*, 2023.