ORAssistant: A Custom RAG-based Conversational Assistant for OpenROAD

Aviral Kaintura*, Palaniappan R†, Shui Song Luar‡, Indira Iyer Almeida‡
*National Forensic Sciences University Delhi Campus, New Delhi, India
† BITS Pilani Hyderabad Campus, Hyderabad, Telangana, India
‡Precision Innovations Inc.
*102ctbmti2122019@nfsu.ac.in, †f20212915@hyderabad.bits-pilani.ac.in, ‡jluar@precisioninno.com, ‡iiyer@precisioninno.com

***Abstract*—Open-source Electronic Design Automation (EDA) tools are rapidly transforming chip design by addressing key barriers of commercial EDA tools such as complexity, costs, and access. Recent advancements in Large Language Models (LLMs) have further enhanced efficiency in chip design by providing user assistance across a range of tasks like setup, decision-making, and flow automation. This paper introduces ORAssistant, a conversational assistant for OpenROAD, based on Retrieval-Augmented Generation (RAG). ORAssistant aims to improve the user experience for the OpenROAD flow, from RTL-GDSII by providing context-specific responses to common user queries, including installation, command usage, flow setup, and execution, in prose format. Currently, ORAssistant integrates OpenROAD, OpenROAD-flow-scripts, Yosys, OpenSTA, and KLayout. The data model is built from publicly available documentation and GitHub resources. The proposed architecture is scalable, supporting extensions to other open-source tools, operating modes, and LLM models. We use Google Gemini as the base LLM model to build and test ORAssistant. Early evaluation results of the RAG-based model show notable improvements in performance and accuracy compared to non-fine-tuned LLMs.**

***Index Terms*—OpenROAD, conversational AI, chatbot, retrieval-augmented generation, electronic design automation, EDA, LLM, RAG, ASIC design, chip design**

I. Introduction

Since 2018, open-source EDA tools have rapidly democratized hardware design and driven innovation through research and collaboration. Free from licensing constraints, they foster a thriving ecosystem of chip design and education [1]–[3]. Recently, ML and GenAI-based chip design methodologies have been applied to open-source tools, yielding significant benefits in productivity [4]. The open infrastructure of these tools simplifies model training, integration and enable the use of shared resources, such as documentation, scripts, and datasets for a host of deployment options [5].

ORAssistant began as a Google Summer of Code (GSoC) 2024 [6] project[1] with an objective of assisting OpenROAD users in completing basic tasks successfully — from setup to flow execution. We focused on addressing frequently occurring problems in areas such as installation, design setup and flow, command usage, and debugging. Traditional user resources, such as documentation and tutorials, tend to become outdated quickly and fail to incorporate practical knowledge gained from collaborative user experiences. Our goal was to build a chatbot that harnesses the dynamic nature of open-source tools and resources to address basic OpenROAD tasks efficiently. We developed a publicly available chatbot [7] that supports continuous improvement and scalability across the tool chain. The complete source code for ORAssistant is available on GitHub [8].

ORAssistant's RAG architecture can be used atop any publicly available LLM. The RAG system enhances the base LLM's output by ensuring that knowledge is retrieved from trusted data sources, generating reliable responses [9]. Our key contributions are:

- **Development of ORAssistant**: A conversational AI agent that assists users in a simple question and answer format with basic conversational abilities.
- **ORAssistant RAG Dataset**: A curated dataset derived from open-source tool documentation and GitHub data from the OpenROAD and OpenROAD-flow-scripts repositories.
- **ORAssistant Evaluation Framework**: Development of a comprehensive evaluation methodology that utilizes both the publicly available EDA Corpus dataset [10] and a self-curated question-answer (QA) dataset. This framework enables robust assessment of LLM performance in the OpenROAD domain, facilitating comparisons between ORAssistant and non-fine-tuned LLMs.

II. Dataset Generation

The effectiveness of any RAG-based system depends on the quality of the underlying data sources, as they form the basis for model accuracy, information retrieval, and response generation. Thus, curating a properly annotated dataset is crucial.

A. Data Sources

ORAssistant primarily assists users of OpenROAD and OpenROAD-flow-scripts while also offering basic support across the RTL-to-GDSII tool chain, from synthesis to layout verification. Its knowledge base includes public documentation, tool manuals, and custom-annotated GitHub discussions, as shown below:

[1]Palaniappan R and Aviral Kaintura contributed equally to this project under the mentorship of Indira Iyer and Jack Luar.

- OpenROAD Documentation [11]
- OpenROAD-flow-scripts Documentation [12]
- OpenROAD Man pages [11]
- OpenSTA Documentation [13]
- Yosys Documentation [14]
- KLayout Documentation [15]
- Research Papers, Tutorials on OpenROAD [16]
- Reformatted and labeled GITHUB discussions [17]

These diverse sources capture detailed information about the tools and their usage. To keep the dataset updated, an automated build script is used to extract and version data from the OpenROAD repository and external sources, ensuring that ORAssistant is equipped with the most relevant and up-to-date information. Additionally, live hyperlinks for each data source are stored, enabling citations during response generation.

B. Issues and Discussion Analysis

Along with public documentation sources, selectively curated conversations from support forums such as GITHUB issues and discussions have been incorporated to identify key problem areas. Using the GITHUB GRAPHQL API, conversations from the issues and discussions pages of the OpenROAD and OpenROAD-flow-scripts repositories, are extracted and stored in JSONL datasets. Since these scraped conversations lack proper annotation, an LLM-based categorization approach has been used to generate three tags for each conversation: category, subcategory, and referenced tools. This characterization enables the correct routing of the tool-based RAG system to domain-specific retriever tools. Detailed analysis indicated that GitHub Issues were predominantly bug reports with limited relevance. GitHub Discussions provided a wider variety of user queries and their corresponding solutions, making it a more valuable data source for ORAssistant.

The current JSONL dataset [18] consists of a total of 736 issues and 344 discussions. Table I summarizes the distribution of issues and discussions by category.

TABLE I
DISTRIBUTION OF GITHUB ISSUES AND DISCUSSIONS BY CATEGORY

Category	Issues (%)	Discussions (%)
Bug	45.90	4.65
Feature request	18.60	8.72
Runtime	13.60	28.50
Build	9.92	8.14
Query	7.34	40.70
Installation	2.85	3.49
Documentation	0.95	1.16
Configuration	0.82	4.65

III. RAG ARCHITECTURE

ORAssistant's tool-based RAG architecture and data ingestion pipeline can be integrated with any LLM, supporting both local and cloud-based deployments. This allows organizations to balance performance requirements, computational resources, and data privacy concerns.

A. Dataset Ingestion

The pipeline processes documents in multiple formats (Markdown, PDF, HTML), and divides them into smaller, manageable document chunks based on format-specific delimiters. These chunks are then fed to a SBERT-based [19] embedding model, which encodes the textual information into dense vector representations. The resulting vectors are then stored in a Facebook AI Similarity Search (FAISS) [20] vector database for downstream use. While this approach efficiently captures the semantic meaning in each chunk, it is unsuitable for retrieving documents based on exact keywords. To perform exact term matching, we use the classical Best Match 25 (BM25) [21] indexing technique, which allows for keyword-based retrieval. The knowledge base is thus represented as a weighted sum of vectors and keyword indices, allowing for both semantic and exact-term matching.

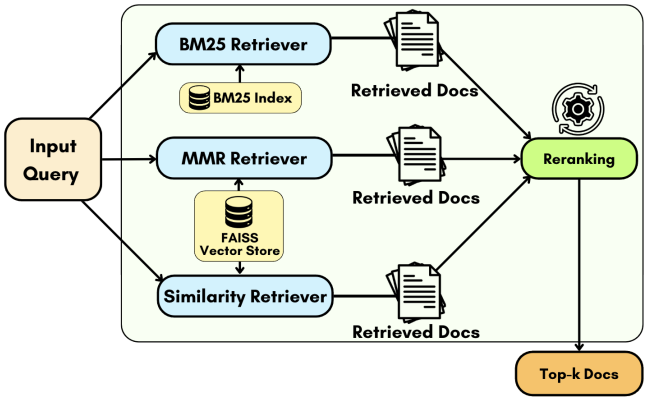

Fig. 1. Hybrid Retriever Function

B. Hybrid Retriever Function

As illustrated in Figure 1, the retriever function searches its input knowledge base to identify document chunks most relevant to a given query. It employs multiple vector search techniques, including similarity search and maximal marginal relevance (MMR) search. In similarity search, chunks with the highest cosine similarity to the input are selected, while MMR search introduces diversity by minimizing redundancy in the retrieved chunks. Additionally, a classical text-based search is performed on the BM25 index to retrieve documents containing the exact keywords specified in the query. A re-ranking model then processes these search results, adjusting the document ranking to provide a top-k selection of the most relevant documents, ensuring both precision and diversity in the final output.

C. Domain Specific Retriever Tools

ORAssistant's knowledge base encompasses a wide range of information from various applications in the OpenROAD flow. Subsets of this knowledge base relevant to specific applications are provided to the hybrid retriever function to form domain specific retriever tools, as listed in Figure 2. A custom prompt guides the base LLM in selecting the

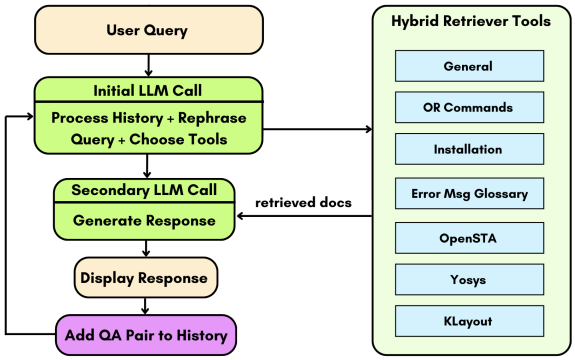

Fig. 2. ORAssistant's tool-based RAG Architecture

appropriate retriever tools for each query. For instance, the *OR Commands* tool retrieves information specific to the OpenROAD framework's commands, while the *Installation* tool focuses on documentation related to installation procedures. In contrast to using a single retriever function on a pooled knowledge base, the modular approach significantly reduces the chances of incorrect document retrieval. Additionally, the architecture allows for future integration with other open-source tools and flow runners within the OpenROAD ecosystem.

D. Context-Aware Response Generation

As the conversation progresses, ORAssistant stores question-answer pairs locally to maintain context. When the user submits a new query, the system first processes the stored conversation history to ensure context continuity. The incoming query is then rephrased, incorporating information from previous exchanges. This gives ORAssistant the capability to answer follow up queries, and thereby maintain long, context-aware conversations.

The tool-based RAG architecture ensures that responses are context-correct by leveraging both the conversation history and domain-specific knowledge sources. These sources enable the system to provide responses with precise citations and hyperlinks for each query. Figure 2 depicts the tool-based RAG system operating in two distinct stages:

- **Initial LLM Call**: ORAssistant processes the conversation history to contextualize and rephrase the user's current query. Based on the refined query, the most appropriate tools are selected and documents are subsequently retrieved.
- **Secondary LLM Call**: ORAssistant generates a response for the rephrased query using the documents retrieved by the selected tools.

IV. Hosted Application

ORAssistant can be accessed on a Next.js [22] based web front-end [7]. The hosted version uses Gemini 1.5 Flash [23] as its base LLM alongside the *text-embedding-004* [24] model for generating vector embeddings. The web application supports the creation of multiple conversation threads, each retaining its own history. This allows users to switch between threads while keeping each discussion focused and relevant to specific topics. Figure 3 shows an ORAssistant-generated, composite response for a user query about floorplan creation options in OpenROAD-flow-scripts.

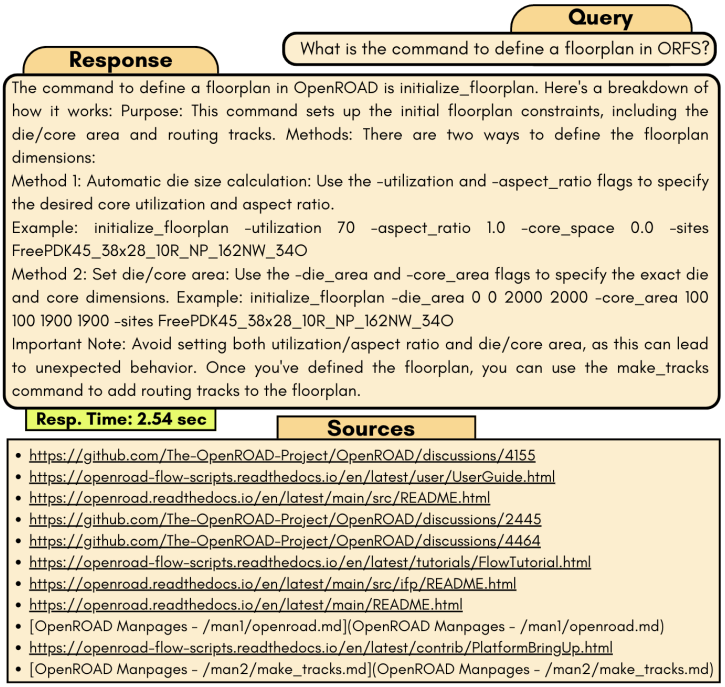

Fig. 3. Example of ORAssistant generated response.

V. Evaluation and Results

The evaluation of ORAssistant is crucial to understanding its performance and limitations. The process aims to quantify the system's ability to successfully retrieve information from correct sources and provide precise responses. The evaluation process identifies areas where the model struggles, allowing for continuous improvement. Additionally, it helps identify gaps in the original documentation and knowledge sources. These insights can guide future improvements through a bidirectional feedback loop.

To assess the effectiveness of our approach, we compared ORAssistant's performance against base pre-trained LLMs like Gemini 1.5 Flash [23] and GPT-4o [25]. We utilize an approach based on GPTScore [26], where a separate LLM is used as an automated evaluator. This *LLM Judge* is given the original question, a ground truth answer, and the model-generated response. Using a carefully designed system prompt, the *LLM Judge* then assesses the quality, coherence, and accuracy of the generated response by comparing it to the ground truth. The evaluation generates the following metrics:

- **Classification metrics**: As shown in Table II, the judge compares the ground truth and LLM's response, classifying them into one of four predefined categories (TP, TN, FP, FN). Metrics such as Accuracy, Precision, Recall, and F1 score are then computed using these classifications.

- **LLMScore**: The judge assigns a score in the range of $[0, 1]$, based on the quality and accuracy of the LLM's response in relation to the ground truth.

TABLE II
EVALUATION METRICS FOR MODEL ANSWERS

Q: What does CTS stand for? **A:** CTS stands for Clock Tree Synthesis. It is a stage... **Eval:** True Positive (TP) (Detailed, accurate, and relevant.)
Q: What is the latest movie released? **A:** I can't provide information on movies... **Eval:** True Negative (TN) (Correctly identified out of scope.)
Q: What does CTS stand for? **A:** CTS stands for Central Time Scheduling... **Eval:** False Positive (FP) (Incorrect and irrelevant.)
Q: What does CTS stand for? **A:** I cannot provide an answer... **Eval:** False Negative (FN) (Failed to answer when expected.)

For our evaluation, we used two QA datasets for evaluation: a custom curated HumanEval dataset with 50 OpenROAD-related QA pairs [27], and 100 QA pairs from the publicly available EDA Corpus dataset [10].

To ensure a fair comparison, we conducted five independent runs for each question across these models: ORAssistant (with Gemini 1.5 Flash), base GPT-4o, and base Gemini 1.5 Flash. Multiple runs help account for the variability in LLM outputs, reducing outliers and improving statistical reliability. Evaluation metrics computed using Gemini 1.5 Pro [28] as the judge LLM have been averaged for each dataset and presented in Table III. As depicted in Table III, ORAssistant significantly outperforms the base pre-trained LLMs across both the EDA Corpus and HumanEval datasets. ORAssistant achieves notably high precision and recall scores, indicating very few false positives and false negatives in its responses. In contrast, both GPT-4o and Gemini 1.5 Flash exhibit subpar performance. Although GPT-4o achieves high recall scores on both datasets, it is offset by a very low precision score. This suggests that the model often hallucinates and generates false positive responses, without acknowledging its lack of knowledge. Across both datasets, ORAssistant attains a considerably high LLMScore, when compared to the base pre-trained LLMs. In terms of response times, ORAssistant averages 2.6 seconds across the testing datasets, while base GPT-4o records 4.7 seconds and base Gemini 1.5 Flash averages 2.3 seconds.

TABLE III
EVALUATION RESULTS ON EDA CORPUS (100 QUESTIONS) AND HUMAN EVAL (50 QUESTIONS) DATASETS.

EDA Corpus Dataset					
Architecture	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	LLMScore (%)
ORAssistant	**90.4**	**94.8**	95.2	**95.0**	**77.7**
GPT-4o	48.4	48.4	**100.0**	65.2	52.6
Gemini 1.5 Flash	38.0	43.3	75.7	55.1	35.7
Human Eval Dataset					
Architecture	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	LLMScore (%)
ORAssistant	**87.2**	**92.4**	94.0	**93.2**	**79.7**
GPT-4o	46.8	46.8	**100.0**	63.8	48.7
Gemini 1.5 Flash	32.8	35.4	80.2	49.1	28.1

Since ORAssistant uses Gemini 1.5 Flash, the contrast between its scores and the base Gemini 1.5 Flash scores highlights how the tool-based RAG architecture guides the LLM towards better performance. While base pre-trained models struggle with relevance due to general training data and outdated knowledge, ORAssistant enhances accuracy by leveraging up-to-date data sources. Moreover, ORAssistant avoids hallucinations by grounding its responses in reliable, deterministic, and contextually relevant data sources.

VI. RELATED WORK

An alternate approach for an OpenROAD Assistant [29] as a chatbot and script generator, uses a fine-tuned LLM. Other tools like the Hybrid RAG based Ask-EDA [30] and the domain-adaptive ChatNeMo [31] use proprietary data. ChatEDA [32] employs fine-tuning for basic task planning in physical design using OpenROAD and other EDA tools. In [33], the authors utilize a RAG based system tailored for OpenROAD and EDA tools, with custom fine-tuned embeddings and reranker models.

RAG offers greater flexibility through real-time adaptation to tool and data source changes. Our tool-based architecture is scalable and aligns well with OpenROAD's modular flow.

VII. FUTURE WORK

Our tool-based architecture enables support for interfacing with OpenROAD's Python-based APIs for custom applications. Combining a fine-tuned model with our RAG architecture will provide dual advantages of real-time adaptability for documentation and enhanced accuracy for tasks like design exploration and script generation. To further enhance the performance of the retriever functions, embeddings and reranker models can be fine-tuned on ORAssistant's knowledge base. Adding human-in-the-loop feedback is another way to continuously improve both the knowledge base and generated responses. We also plan to interface ORAssistant with OpenROAD command-line interface (CLI) and graphical-user-interface (GUI).

VIII. CONCLUSION

In this paper, we present ORAssistant, a RAG-based assistant built using a scalable, tool-based architecture for the OpenROAD flow. This chatbot assists users by providing contextual and reliable answers for common user queries using native and publicly available data sources. Initial results show that the RAG model outperforms base pre-trained LLMs from metrics derived from human evaluations and automated LLM-based methods. ORAssistant showcases the potential of GenAI-based tools in chip design to enhance user experience, enabling users to learn faster and gain deeper insights across all levels of expertise. This work, supported by the OpenROAD project team, GSoC, and OSRE, is the result of collaborative and open-source community-driven EDA advocacy, and contribution.

REFERENCES

[1] A. B. Kahng, "Open-source EDA: If we build it, who will come?" in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*. IEEE, 2020, pp. 1–6.

[2] Efabless. ChipIgnite. Accessed: 2024-09-22. [Online]. Available: https://efabless.com/chipignite

[3] M. Venn, "Tiny Tapeout: A shared silicon tape out platform accessible to everyone," *IEEE Solid-State Circuits Magazine*, vol. 16, no. 2, pp. 20–29, 2024.

[4] Efabless. (2023) Using Generative AI for ASIC Design. Accessed: 2024-09-22. [Online]. Available: https://www.youtube.com/watch?v=-R7limdUyts

[5] AWS. (2023) Open source chip design on AWS. Accessed: 2024-09-22. [Online]. Available: https://aws.amazon.com/blogs/industries/open-source-chip-design-on-aws/

[6] U. S. C. O. S. P. Office. (2024) Open Source Research Experience (OSRE 2024). Accessed: 2024-09-22. [Online]. Available: https://ucsc-ospo.github.io/osre24

[7] The-OpenROAD-Project. (2024) ORAssistant front end. Accessed: 2024-09-22. [Online]. Available: https://orassistant.netlify.app/

[8] The OpenROAD Project, "ORAssistant," 2024. [Online]. Available: https://github.com/The-OpenROAD-Project/ORAssistant

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[10] B.-Y. Wu, U. Sharma, S. R. D. Kankipati, A. Yadav, B. K. George, S. R. Guntupalli, A. Rovinski, and V. A. Chhabria, "EDA Corpus: A large language model dataset for enhanced interaction with OpenROAD," *arXiv preprint arXiv:2405.06676*, 2024.

[11] The-OpenROAD-Project. (2024) OpenROAD Documentation. Accessed: 2024-09-22. [Online]. Available: https://openroad.readthedocs.io

[12] ——. (2024) OpenROAD-flow-scripts documentation. Accessed: 2024-09-22. [Online]. Available: https://openroad-flow-scripts.readthedocs.io

[13] ——. (2024) OpenSTA documentation. Accessed: 2024-09-22. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenSTA/tree/master

[14] YosysHQ. (2024) YosysHQ documentation library. Accessed: 2024-09-22. [Online]. Available: https://yosyshq.readthedocs.io

[15] K. Project. (2024) KLayout documentation. Accessed: 2024-09-22. [Online]. Available: https://www.klayout.de/doc.html

[16] The-OpenROAD-Project. (2024) The-OpenROAD-Project. Accessed: 2024-09-22. [Online]. Available: https://theopenroadproject.org/

[17] ——. (2024) OpenROAD repository. Accessed: 2024-09-22. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD

[18] ——. (2024) ORQA RAG dataset huggingface repository. Accessed: 2024-09-22. [Online]. Available: https://huggingface.co/datasets/The-OpenROAD-Project/ORQA_RAG_datasets

[19] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[20] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The FAISS library," *arXiv preprint arXiv:2401.08281*, 2024.

[21] E. A. Stuart, "Matching methods for causal inference: A review and a look forward," *Statistical science: a review journal of the Institute of Mathematical Statistics*, vol. 25, no. 1, p. 1, 2010.

[22] Vercel. (2024) Next.js - the react framework. Accessed: 2024-09-22. [Online]. Available: https://nextjs.org/

[23] G. DeepMind. (2024) Gemini flash. Accessed: 2024-09-22. [Online]. Available: https://deepmind.google/technologies/gemini/flash/

[24] Google Cloud. (2024) Text embeddings API. Accessed: 2024-09-22. [Online]. Available: https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/text-embeddings-api

[25] OpenAI. (2024) Hello gpt-4o. Accessed: 2024-09-22. [Online]. Available: https://openai.com/index/hello-gpt-4o/

[26] J. Fu, S.-K. Ng, Z. Jiang, and P. Liu, "GPTScore: Evaluate as you desire," *arXiv preprint arXiv:2302.04166*, 2023.

[27] The-OpenROAD-Project. (2024) ORAssistant RAG dataset. Accessed: 2024-09-22. [Online]. Available: https://huggingface.co/datasets/The-OpenROAD-Project/ORAssistant_HumanEval_QA_Pairs

[28] Google DeepMind. (2024) Gemini pro. Accessed: 2024-09-22. [Online]. Available: https://deepmind.google/technologies/gemini/pro/

[29] U. Sharma, B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria, and A. Rovinski, "OpenROAD-Assistant: An open-source large language model for physical design tasks," in *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*, 2024, pp. 1–7.

[30] L. Shi, M. Kazda, B. Sears, N. Shropshire, and R. Puri, "Ask-EDA: A design assistant empowered by LLM, hybrid RAG and abbreviation de-hallucination," *arXiv preprint arXiv:2406.06575*, 2024.

[31] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu *et al.*, "ChipNeMo: Domain-adapted LLMs for chip design," *arXiv preprint arXiv:2311.00176*, 2023.

[32] H. Wu, Z. He, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu, "ChatEDA: A large language model powered autonomous agent for EDA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[33] Y. Pu, Z. He, T. Qiu, H. Wu, and B. Yu, "Customized retrieval augmented generation and benchmarking for EDA tool documentation QA," *arXiv preprint arXiv:2407.15353*, 2024.