OpenROAD-Assistant: An Open-Source Large Language Model for Physical Design Tasks

Utsav Sharma*[1], Bing-Yue Wu*[2], Sai Rahul Dhanvi Kankipati[2], Vidya A. Chhabria[2], and Austin Rovinski[1]
[1]New York University; [2]Arizona State University

***Abstract*—Large language models (LLMs) have shown significant potential in serving as domain-specific chatbots. Recently, these models have emerged as powerful tools for chip design, providing both natural language responses and script generation for domain-specific inquiries. Previous work has demonstrated the effectiveness of LLMs in assisting with physical design automation; however, these approaches often rely on proprietary tools, APIs, technologies, and designs. As a result, access to these models is extremely limited, particularly for new chip designers who could greatly benefit from a design assistant. This paper introduces OpenROAD-Assistant, an open-source chatbot for OpenROAD that relies only on public data and responds to queries in either prose or Python script using the OpenROAD APIs. OpenROAD-Assistant leverages the Llama3-8B foundation model and employs retrieval-aware fine-tuning (RAFT) to respond to physical design-specific questions for OpenROAD. Notably, OpenROAD-Assistant outperforms other foundational models such as ChatGPT3.5, ChatGPT4, Code Llama, Claude3, and other ablation study baselines on the measured metrics (pass@k for scripting and BERTScore/BARTScore for question-answering). OpenROAD-Assistant achieves a 77% pass@1 score, 80% pass@3 score for scripting, and it achieves a 98% BERTScore and 96% BARTScore on question-answering.**

I. Introduction

Physical design of digital integrated circuits (ICs) is a complex multi-stage process that demands extensive domain expertise in both chip design and the use of electronic design automation (EDA) tools. The necessary tool proficiency not only creates large barriers to entry for newcomers to the field, but also creates productivity challenges for experienced chip designers who transition to a new EDA tool or to an unfamiliar stage of the physical design flow. These challenges result in decreased productivity and extended design schedules.

Recent works on domain-specific LLMs for chip design [1]–[4] have shown that tailored datasets and fine-tuning mechanisms can significantly improve performance over foundation models such as [5], [6]. In comparison to models related to the generation of frontend register transfer-level (RTL) design [3], [4] which have significant data available due to open-source languages such as Verilog, models focused on backend physical design are heavily limited in training data due to a reliance on proprietary APIs of EDA tools, technologies, and designs.

To date, LLM methodologies in physical design EDA have faced significant challenges due to the absence of accessible, open infrastructure. The high cost of licenses for access to commercial EDA tools, documentation, and tutorials places them out of reach for the wider community. Moreover, these tools are bound by strict end-user license agreements which often restrict activities such as benchmarking, training AI with their documentation, and freely exchanging user scripts.

Training an accessible LLM for chip design results in a bootstrapping problem. Current models require ingestion of large amounts of data in order to be useful, but large amounts of open-source data do not exist. Recent efforts such as EDA Corpus [7], [8] have created a small set of open-source supervised training data for OpenROAD [9], but this data ($<$600 unique data pairs) is orders of magnitude less than what prior approaches have used to demonstrate success.

In this work, we present OpenROAD-Assistant, an open-source LLM chatbot for OpenROAD trained with EDA Corpus [7], [8]. OpenROAD-Assistant is trained to generate script and question-answer responses. It uses quantized low-rank adaptation (QLoRA), to develop two separate adapters for prompt-script and question-answer data. oBoth adapters use retrieval-augmented fine-tuning (RAFT) [10] which combines the strengths from both retrieval augmented generation (RAG) and supervised fine-tuning (SFT) to improve the accuracy of the chatbot even with limited training data. OpenROAD-Assistant uses Llama3 as a foundation model for prompt-response and mxbai-embed-large-v1 for retrieval. It relies on open-source EDA tool source code, documentation, and open-source supervised training dataset to develop the model and RAG database to answer OpenROAD-related questions and generate tool scripts.

For the question-answer (QA) adapter, we perform SFT using EDA Corpus and for the script generation adapter, we perform fine-tuning using templated scripts and their descriptions from EDA Corpus. Unlike prior work [1], we do not perform domain-adaptive pre-training (DAPT) on our foundation models using large amounts of data. Instead, for the QA adapter, we use additional stages of SFT for instruction models and data retrieval using public datasets. In addition, we create an automated generator of an SFT layer for further fine-tuning the QA adapter on paraphrased retrieved data, which improves the quality of the generated response.

Our key contributions include the following:

- Development of a novel open-source physical design chatbot for script generation and question-answering using OpenROAD
- Curation and release of a physical design RAG database for script generation and QA

* denotes equal contribution

- Significant improvements in physical QA and script generation over state-of-the-art foundation models
- An ablation study demonstrating performance improvements of our techniques and various fine-tuning baselines

For script generation, out model achieves pass@1 and pass@3 scores of 77% and 80%, respectively, outperforming untuned commercial baselines (3% and 13%) and commercial baselines finetuned with EDA Corpus (20% and 37%). For question-answering, our model also achieves a BERTScore F1 [11] and BARTScore F1 [12] of 98% and 96%, respectively. These scores significantly outperform untuned commercial baselines (85% and 57%) and commercial baselines finetuned with EDA Corpus (84% and 54%).

OpenROAD-Assistant, the RAG database, and its training and inference scripts are available under an anonymized repository for review at [13]. To the best of our knowledge, this is the first publicly released chatbot designed for physical design tasks, including question-answering and script generation. With the release of this model, we aim to initiate efforts toward training LLM assistants specifically for physical design. We envision that the model will benefit OpenROAD users and serve as a benchmark for developing chatbots for EDA tools.

II. Related Works

LLMs for frontend RTL design and synthesis Recently, several works have been exploring the use of LLMs in chip design. Many works focus on high-level design such as for high-level synthesis (HLS) [14], RTL generation and verification [1], [3], [15]–[19], and chatbot assistants for these topics [1]. These works all strongly benefit from open-source languages such as Verilog and VHDL, which enable access to significant amounts of online training data and open-sourcing designs. Prior works have even shown that state-of-the-art foundation models can perform well at RTL code generation without any domain specialization [4].

LLMs for backend physical design When transitioning to physical design automation, however, the amount of publicly available data for training drops dramatically. Physical design automation is dominated by proprietary tool APIs and technologies, which necessarily means that the data cannot be publicly accessed for training. There have been two recent prior works that have used LLMs for physical design:

(i) ChipNeMo [1]: A recent work that leverages NVIDIA proprietary designs, documentation, scripts, etc., to produce a design assistant capable of several design tasks, including answering questions, documentation questions, and tool script generation. While this model performs very well on these tasks, the model and dataset remain publicly inaccessible.

In contrast, OpenROAD-Assistant is trained on less amounts of tokens from open-source data, capable of answering both physical design questions and script generation. Instead of using domain adaptive training on the foundational model as ChipNeMo does, our work employs a simpler, yet efficient RAFT technique which adapts OpenROAD-Assistant to generate only domain-specific answers. Our work only uses a curated dataset of high-quality, an approach that aligns with the findings presented in [20] which shows that high-quality, smaller datasets can yield highly effective models.

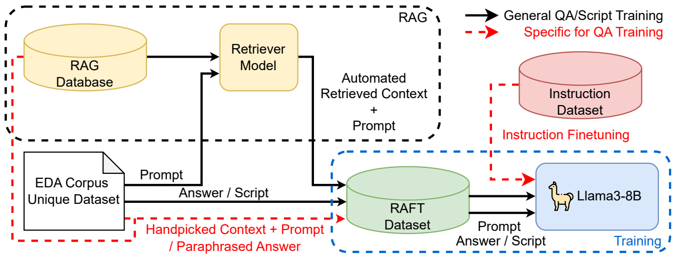


Fig. 1. OpenROAD-Assistant training flow

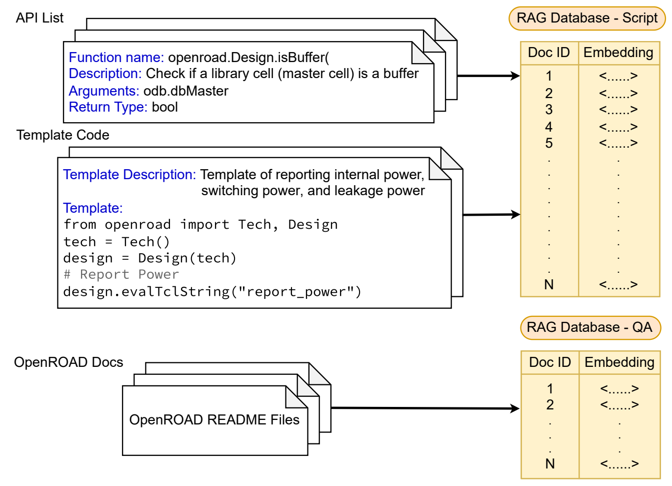
(ii) ChatEDA [2]: In contrast to ChipNemo, ChatEDA trains a model to generate EDA tool scripts for open-source physical design tools including OpenROAD [9], [21]. However, ChatEDA focuses solely on automation script generation and physical design flow task planning (sequence of physical design stages). While some of the underlying training data is open-source [22], the model has not been publicly released and only a small subset of the training data has been publicly released, making the model difficult to reproduce and access. Further, ChatEDA is trained to generate high-level APIs for physical design flow tasks that further require translation to specific EDA tool APIs limiting the interaction between the user and the LLM only those APIs that have translators. The use of high-level APIs simplifies the problem of script generation but requires translators. In our work, we develop scripts for direct EDA tools without the need of translators. We also show in Section IV that if OpenROAD-Assistant worked on the simplified problem of only generating the top-level APIs, the training methodologies proposed in this work generate higher accuracy than that of ChatEDA.

In contrast to ChatEDA, our work performs both script generation and question-answering for physical design tasks. OpenROAD-Assistant training dataset is open-source, the model is open-source and available for the public to use. The generated scripts directly query, change the database and even perform physical design flow stages and task planning for OpenROAD. OpenROAD-Asisstant leverages the publically available OpenROAD source code to train the model and can work with limited dataset (400 datapoints) unlike ChatEDA that requires over 1500 datapoints.

The underlying theme of the related work is that sufficient public data for physical design is *extremely scarce* and it makes training effective physical design LLMs difficult. Furthermore, no work to date has released the full training set and model for physical design, meaning that those who have potentially the most to benefit from physical design assistants – chip design students and enthusiasts – do not have access to the models. OpenROAD-Assistant aims to fill this gap.

III. OpenROAD-Assistant Model Architecture

The training flow for OpenROAD-Assistant is shown in Fig. 1. OpenROAD-Assistant employs quantized low-rank adaptation (QLoRA) to create separate model adapters for prompt-script


API List
Function name: openroad.Design.isBuffer(
Description: Check if a library cell (master cell) is a buffer
Arguments: odb.dbMaster
Return Type: bool
Template Code
Template Description: Template of reporting internal power, switching power, and leakage power
Template:
RAG Database - Script
Doc ID
Embedding
RAG Database - QA
OpenROAD Docs
OpenROAD README Files

```
from openroad import Tech, Design
tech = Tech()
design = Design(tech)
# Report Power
design.evalTclString("report_power")
```

Doc ID	Embedding
1	<......>
2	<......>
3	<......>
4	<......>
5	<......>
.	.
N	<......>

Doc ID	Embedding
1	<......>
2	<......>
.	.
N	<......>

Fig. 2. OpenROAD-Assistant RAG Database for both Prompt-Script and Question-Answer adapters

and question-answer data. As such, the training flow in Fig. 1 is performed separately on each dataset. While the training flows are similar for both adapters, we performed additional steps for the question-answer training flow (marked in red).

To train the system, we start by querying the retriever model with queries in the training set. The retriever model then searches through the database, which includes 1) an OpenROAD API list plus code templates for the prompt-script dataset or 2) OpenROAD documentation for the question-answer dataset. Then, the retrieved context is merged into the supervised data to create a RAFT dataset. Lastly, the RAFT dataset is used to fine-tune the llama3-8B foundation model [6]. In the case of question-answering, two additional steps are performed: 1) Answers in the RAFT dataset are paraphrased to help reduce direct copying from the provided context to the final output. 2) We perform an initial SFT using an instruction fine-tuning dataset to aid question-answering.

A. Retriever Model

OpenROAD-Assistant uses RAG in order to retrieve relevant contexts from a domain-specific database. RAG has shown to be extremely useful to provide high recall on low amounts of source data by embedding the data directly into a vector format rather than using the data for fine-tuning. Because RAG does not require fine-tuning, it reduces the computational overhead of introducing new data to the model which is useful for source data which changes fairly often.

Fig. 2 demonstrates how the RAG database for each adapter is created. For the code-script database, we create two types of data. One is an API list, which is a set of OpenROAD functions used to query or modify the OpenROAD design database. The other is a set of code templates which describe a set of common actions, particularly for EDA flow stage implementations. For example, using OpenROAD Python scripts to perform global routing. For the question-answer database, we also include two types of data. The first is the set of publicly available OpenROAD documentation, including READMEs, manpages, and other documentation. The second is the QA training set taken from EDA Corpus. Datapoints from the test set are not included in the RAG database. These source data are then embedded into a vector format which is accessible by the retrieval model. This model will search the database for the k most relevant contexts and return them as output.

B. Retrieval Augmented Fine-Tuning (RAFT)

Using the retriever model, we perform RAFT [10]. This technique has shown to improve the accuracy and reduce hallucinations when the retriever model provides out-of-context data. RAFT essentially fine-tunes a foundation model by adding context to a supervised dataset, which enables the model to better discriminate against irrelevant context provided by the retriever model. In our system, we use RAFT on both prompt-script and question-answer adapters. We find that RAFT is useful in discriminating against unhelpful APIs which are returned by the retriever model and can cause incorrect commands to be used in the generated script. Additionally, RAFT improves the accuracy of question-answer responses by discriminating against unrelated documentation. Without RAFT, we find that the models become overly dependent on the accuracy of the retrieved context.

C. Supervised Fine-Tuning (SFT)

For the question-answer adapter, we first perform SFT for the foundation model on an instruction dataset, which is a generalized dataset for performing question-answering tasks. The purpose of this is to leverage publicly available data to train the model for the designated task of question-answering, even if the dataset does not contain EDA-specific information.

We perform this task to enhance the effective amount of training data for the question-answering task and we found that it improved the evaluated scores in Section IV. Prior work has also noted that instruction fine-tuning can improve results on domain-specific tasks [23], [24]. This fine-tuning is not performed for the prompt-script adapter, because we were unable to find a suitable public dataset which enhances the overall accuracy of the model.

After instruction SFT, we then perform SFT using the RAFT dataset. For the question-answer model, we perform automated paraphrasing of answers in the RAFT dataset using GPT-4 prior to training. This serves two purposes: 1) help reduce direct copying from context to output and 2) improve semantic understanding. We found that this can offer some minor improvements by reducing the foundation model's dependence on the semantics and phrasing of the provided context. In the prompt-script model, we do not perform this step because code is more difficult to automatically paraphrase and it may not be as helpful as prose.

D. Inference Flow

Fig. 3 depicts the flow of prompting OpenROAD-Assistant. During inference, questions related to OpenROAD script generation or general OpenROAD questions are passed to the retriever model to be embedded with top-n relevant context from the database, are then passed to the OpenROAD-Assistant with the corresponding adapter to generate the answer.

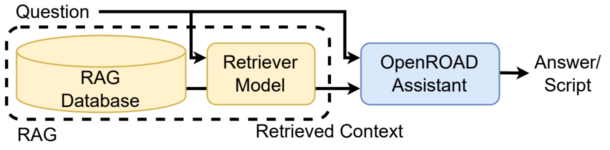


Fig. 3. OpenROAD-Assistant inference flow.

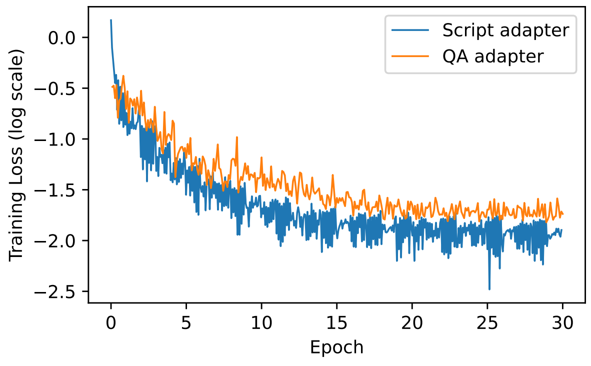


Fig. 4. Training loss converging for script adapter and QA adapter.

IV. Evaluation of OpenROAD-Assistant

A. Methodology

For our retrieval model, we use mxbai-embed-large-v out-of-the-box with no additional training. We found that the model performed well at context retrieval without SFT, and we additionally found that SFT did not significantly improve context retrieval. For our RAG database, we use an open-source sentence embedding model [25] to vectorize our dataset retrieve the best n documents where we select $n = 3$. The code and question-answer databases are each used only for their respective adapter.

The foundation model used is the 4-bit quantization of Llama3-8B [6], which significantly improves the model's memory footprint and computational requirements over higher-precision models, especially on hardware with limited resources. To fine-tune Llama3-8B, we used QLoRa [26] and RAFT as discussed in Section III-B. The hyperparameters parameters used during fine-tuning are shown in Table I. In the case of the prompt-script (PS) adapter, only one round of SFT was applied. For the question-answering (QA) adapter, instruction SFT (Section III-C) was applied first and then OpenROAD question-answering SFT was applied. We use the SlimOrca [27] dataset for instruction SFT.

It takes 2 hours and 37 minutes to train the OpenROAD-Assistant's script adapter using four NVIDIA RTX A5500 GPUs, and it takes 50 minutes to train the OpenROAD-Assistant's Question-answer adapter using one NVIDIA V100 GPU. OpenROAD-Assistant is trained for 30 epochs until the training loss converges, as shown in Fig. 4. The figure plots the loss in log scale for 30 epochs during training for both QA and script generation adapters.

We evaluate OpenROAD-Assistant on a randomly sampled testset from the non-augmented version of EDA Corpus [8].

TABLE I
Hyperparameters used for tuning OpenROAD-Assistant

Hyperparameter	PS adapter	QA adapter	Inst. SFT
Epochs	30	30	1
Learning rate	1.0E-4	1.0E-4	2.0E-4
Quantization	4-bit	4-bit	4-bit
LoRA alpha	16	16	16
LoRA dropout	0	0	0
Bias	None	None	None
Train batch size per GPU	1	2	2
Gradient accumulation step	1	4	4

We compare the OpenROAD-Assistant's responses against baseline foundation models, including Claude3 Sonnet [28], ChatGPT-4 Omni [5], and Llama3 (8B parameter model) [6]

We also perform the following ablation studies to evaluate the impact of our proposed training methodology:

1) Fine-tuned (FT) Llama3 and FT GPT-3.5 Turbo: We evaluate the responses of fine-tuned Llama3 and ChatGPT-3.5 Turbo using EDA Corpus dataset [7], [8].
2) RAG FT Llama3: We implement RAG when performing inference on the fine-tuned Llama3 (i) to compare the performance of the model with and without RAG. This experiment is crucial as it highlights the importance of the creation of RAG database which is only possible using the open-source EDA tool source code and documention.
3) RAFT Code Llama2: For script generation only, we use the same RAFT-based training methodology as OpenROAD-Assistant but use Code Llama2 as a foundation model instead of Llama3.

We evaluate script generation adapters and QA adapters separately using different metrics as described below.

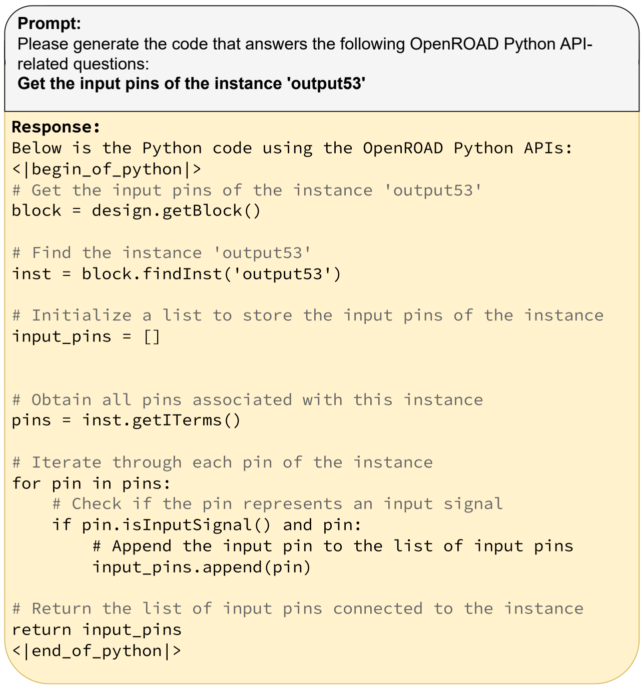
B. Evaluation of Script Generation

Metrics For our testset for script generation, we use 30 datapoints randomly sampled and separated from the non-augmented prompt-script dataset of EDA Corpus [8]. We evaluate the responses generated by OpenROAD-Assistant using a pass@k approach [29]. The pass@k metric evaluates the generated code by executing it and checking whether all the test cases are passed. However, this binary approach, which solely assesses whether the code is completely correct, hinders quality evaluation at a more granular level. Therefore, in combination with pass$k$ we also categorize the generated scripts into correct, partially correct, as explained below:

1) Correct: A response is classified as correct when one of the k generated scripts can run in OpenROAD without errors and generates the same result as our regression test from the golden script.
2) Partially correct (Partial): A response is classified as partially correct if it does not belong to the "Correct" category, and if one of its k responses requires a change in less than one line of code to fall into the "Correct" category due to the generation of a hallucinated "synonym" API in the script.

TABLE II
COMPARING OPENROAD-ASSISTANT AGAINST FOUNDATION MODELS AND ABLATION BASELINES FOR SCRIPT GENERATION (LEFT) AND QUESTION-ANSWERING (RIGHT) USING METRICS AND CATEGORIES DEFINED IN SECTION IV.

Category or Metrics		**Script adapter**						**QA adapter**					
		pass@1			**pass@3**			**BERTScore**			**BARTScore**		
		Correct	Partial	Wrong	Correct	Partial	Wrong	Precision	Recall	F1	Precision	Recall	F1
Foundation models	**Claude3**	0	2	28	0	3	27	0.8169	0.8337	0.8251	0.4681	0.5091	0.4855
	GPT-4 Omni	1	1	28	4	3	23	0.8558	0.8549	0.8552	0.5634	0.5778	0.5689
	Llama3	0	0	30	0	0	30	0.8354	0.8454	0.8403	0.5222	0.5401	0.5301
Ablation studies	**FT GPT-3.5 Turbo**	6	6	18	11	4	15	0.8441	0.8441	0.8435	0.5514	0.5489	0.5393
	FT Llama3	13	4	13	14	3	13	0.8568	0.8463	0.8514	0.5899	0.5576	0.5714
	RAG FT Llama3	17	5	8	19	5	6	0.8690	0.8641	0.8655	0.6206	0.6573	0.7099
	RAFT Code Llama	20	3	7	20	3	7	-	-	-	-	-	-
OpenROAD-Assistant		**23**	**6**	**1**	**24**	**5**	**1**	**0.9831**	**0.9855**	**0.9843**	**0.9517**	**0.9604**	**0.9557**


Prompt:
Please generate the code that answers the following OpenROAD Python API-related questions:
Get the input pins of the instance 'output53'

Response:
```
Below is the Python code using the OpenROAD Python APIs:
<|begin_of_python|>
# Get the input pins of the instance 'output53'
block = design.getBlock()

# Find the instance 'output53'
inst = block.findInst('output53')

# Initialize a list to store the input pins of the instance
input_pins = []


# Obtain all pins associated with this instance
pins = inst.getITerms()

# Iterate through each pin of the instance
for pin in pins:
    # Check if the pin represents an input signal
    if pin.isInputSignal() and pin:
        # Append the input pin to the list of input pins
        input_pins.append(pin)

# Return the list of input pins connected to the instance
return input_pins
<|end_of_python|>
```

Fig. 5. Example of OpenROAD-Assistant's script generation response.

3) Wrong: If none of the k responses fall into either a "Correct" or "partially correct" category.

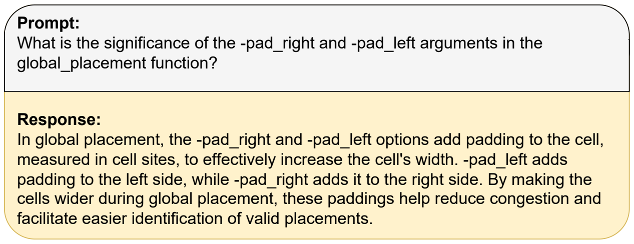
The partially correct responses are still useful for script generation as several EDA tools are capable of correcting users with synonym API. For instance, if the generated model uses a "check_timing" API instead of a "report_checks" API, the tools can respond with a "Did you mean report_checks?" which allows the user to update the generated Partially correct script to a fully Correct script.

Evaluation Using the above metrics, we evaluate OpenROAD assistant's script generation and compare the responses against the foundation models listed in Table II. The table highlights the number of correct, partially correct, and wrong responses from each model for pass@1 and pass@3 for the 30 prompts in the testset. We find that OpenROAD-Assistant outperforms Claude3, ChatGPT-4 Omni, and Code Llama foundation models in terms of correctness of the script generated. Of the 30 testcases, OpenROAD-Assistant generated 23 correct responses for pass@1 and 24 correct responses for pass@3 compared to the only handful of correct responses by the direct application of the foundation models.

For the three ablation studies described in Section IV-A, we find that using our proposed RAFT-based training approach had the most Correct responses on our testset compared to other ablation experiments with an 80% (24/30) pass@3 score. Of the 30 test prompts, OpenROAD-Assistant correctly generated 24, five were partially correct (change in one line of code due to a synonym API), and only one was wrong. Fine-tuning a foundation model alone results in accuracies of 36% (GPT3.5 Turbo) and 50% (Llama3) for pass@3. Fine-tuning Llama3 with RAG-based inference improves the accuracy to 63% which shows the impact of the creation of the RAG database. Further, a RAFT-based training of Code Llama which resulted in 66% accuracy.

An example response to a prompt provided to OpenROAD-Assistant trained using RAFT is shown in Fig. 5. The prompt requests a list of all input pins of a specific instance. OpenROAD-Assistant generates the script between the "begin of python" and "end of python" tokens.

Comparison with prior work In comparison to ChatEDA [2], which generates top-level APIs for physical design flow steps and depends on translators to convert the APIs to EDA tool script, OpenROAD-Assistant generates EDA tool scripts directly as shown in Fig. 5. The problem that ChatEDA solves is a simpler problem compared to the problem OpenROAD-Assistant solves as ChatEDA only performs task planning and generates a few lines of code corresponding to that task. In contrast, OpenROAD-Assistant performs task planning generates several lines of codes related to the tasks, and generates logic-heavy code for interactions with the EDA tool database. While these are qualitative differences between ChatEDA and OpenROAD-Assistant, we still try to perform numerical comparisons by evaluating OpenROAD-Assistant's training methodology on the simpler task planning and generating few lines of code related to the task. We find that using OpenROAD-Assistant's training method and model, task planning, and script generation is 93% accurate for pass@1



Prompt:
What is the significance of the -pad_right and -pad_left arguments in the global_placement function?

Response:
In global placement, the -pad_right and -pad_left options add padding to the cell, measured in cell sites, to effectively increase the cell's width. -pad_left adds padding to the left side, while -pad_right adds it to the right side. By making the cells wider during global placement, these paddings help reduce congestion and facilitate easier identification of valid placements.

Fig. 6. Example of OpenROAD-Assistant's general question response.

and pass@3, while ChatEDA was reported to be 88% accurate.

C. Evaluation of Question-Answering

Similar to the evaluation of our script generation, we use 20 randomly sampled data points from the Q&A dataset of EDA Corpus. To evaluate the responses generated by OpenROAD-Assistant for question-answering, we leverage two metrics:
BERTScore: An evaluation metric designed to assess the quality of text by leveraging the deep contextual representations provided by models like BERT [11]. Unlike traditional metrics which rely on exact word overlaps, BERTScore evaluates the semantic similarity between texts using contextual embeddings from BERT or similar models. By using BERT's deep contextual embeddings, BERTScore can effectively capture semantic meanings beyond mere lexical matches, which is beneficial for evaluating tasks like translation, summarization, or any text generation task where nuanced understanding is crucial.
BARTScore: An evaluation metric developed to evaluate generated text by leveraging the capabilities of the BART model, a pre-trained sequence-to-sequence model [12]. Unlike traditional metrics that often focus on surface-level similarities or exact lexical matches, BARTScore conceptualizes the evaluation of generated text as a text generation problem itself.

Table II shows the results for OpenROAD-Assistant and baseline models on the question-answering task. The models are evaluated using Precision, Recall, and F1 score derived from both BERTScore and BARTScore using the supervised answer as the optimal response. Our proposed model achieves over 98% in precision, recall, and F1 score for BERTScore and over 95% for the same categories for BARTScore. OpenROAD-Assistant outperforms the baseline foundation models Claude3, GPT-4 Omni and Llama3 which achieve only 85% for BERTScore. For BARTScore, our model achieves over 95% in each of the three categories whereas the foundation models perform only achieve 57% in the best case.

Furthermore, we perform an ablation study as mentioned in Section IV-A. Fine-tuning of GPT-3.5 Turbo and Llama3 result in modest improvements to the BERTScore and BARTScore. FT GPT-3.5 performs about as well as GPT-4 Omni (84% vs. 86% F1 BERTScore; 54% vs 57% BARTScore), and FT Llama3 improves modestly over Llama3 (84% to 85% F1 BERTScore; 53% to 57% F1 BARTScore;). The addition of RAG modestly improves BERTScore (85% to 87% F1), but has a significant improvement in BARTScore (57% to 71% F1). Our model which additionally incorporates RAFT, paraphrasing, and instruction SFT further improves the BERTScore (87% to 91% F1) and BARTScore (71% to 96%) to achieve the best scores in all categories. We do not evaluate the question-answering dataset with RAFT Code Llama due to it being a code-based model. An example response from OpenROAD-Assistant is shown in Fig. 6 for a datapoint from our testset.

D. Evaluation of a Combined Model

We also evaluated OpenROAD-Assistant by training a single adapter on both the prompt-script and question-answer datasets using the code-only flow from Fig. 1. For the combined adapter, we found that the adapter performed worse on both scripting and QA tasks than separately trained adapters. We also found that applying the supplemental techniques for question-answering improved the QA scores but at the cost of worse script generation scores. For the single adapter, the model achieved pass@1 score of 19, and pass@3 score of 22. However, the BERTScore F1 was 0.8501 and the BARTScore F1 was 0.7267 for QA.

V. Conclusion

In this paper, we presented OpenROAD-Assistant, an OpenROAD chatbot that can answer prompt-script question-answer queries for physical design. To our knowledge, OpenROAD-Assistant is the first open-source, publicly accessible EDA chatbot which can serve as an excelent starting point for both new hardware designers trying physical design for the first time or seasoned engineers learning OpenROAD's API and flow. Our evaluation shows that the OpenROAD-Assistant significantly outperforms other LLM baselines in all measurements: Precision, Recall, and F1 for BERTScore (98%) and BARTScore (96%) on question-answering tasks, and pass@1 (77%) and pass@3 (80%) for scripting tasks. The model is open-source and available at [13].

References

[1] M. Liu, *et al.*, "ChipNeMo: Domain-Adapted LLMs for Chip Design," *arXiv preprint arXiv:2311.00176*, 2024.
[2] H. Wu, *et al.*, "ChatEDA: A Large Language Model Powered Autonomous Agent for EDA," *IEEE T. Comput. Aid. D.*, 2024.
[3] M. Liu, *et al.*, "VerilogEval: Evaluating Large Language Models for Verilog Code Generation," in *Proc. ICCAD*, 2023.
[4] J. Blocklove, *et al.*, "Chip-Chat: Challenges and Opportunities in Conversational Hardware Design," in *MLCAD*, 2023.
[5] "ChatGPT." https://openai.com/chatgpt, 2023.
[6] AI@Meta, "Llama3." https://github.com/meta-llama/llama3, 2024.
[7] B.-Y. Wu, *et al.*, "Eda corpus: A large language model dataset for enhanced interaction with openroad," 2024.
[8] "EDA Corpus." https://github.com/OpenROAD-Assistant/EDA-Corpus, 2024.
[9] T. Ajayi, *et al.*, "INVITED: Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project," in *Proc. DAC*, pp. 1–4, 2019.
[10] T. Zhang, *et al.*, "RAFT: Adapting Language Model to Domain Specific RAG," *arXiv preprint arXiv:2403.10131*, 2024.
[11] T. Zhang, *et al.*, "Bertscore: Evaluating text generation with bert," 2020.
[12] W. Yuan, *et al.*, "Bartscore: Evaluating generated text as text generation," *ArXiv*, vol. abs/2106.11520, 2021.
[13] "OpenROAD-Assistant." https://github.com/OpenROAD-Assistant/OpenROAD-Assistant, 2024.
[14] L. J. Wan, *et al.*, "Software/Hardware Co-design for LLM and Its Application for Design Verification," in *Proc. ASP-DAC*, 2024.
[15] Y. Tsai, *et al.*, "RTLFixer: Automatically Fixing RTL Syntax Errors with Large Language Models," *arXiv preprint arXiv:2311.16543*, 2024.
[16] M. Orenes-Vera, *et al.*, "From RTL to SVA: LLM-assisted Generation of Formal Verification Testbenches," *arXiv preprint arXiv:2309.09437*, 2023.

[17] X. Meng, *et al.*, "Unlocking Hardware Security Assurance: The Potential of LLMs," *arXiv preprint arXiv:2308.11042*, 2023.
[18] S. Thakur, *et al.*, "AutoChip: Automating HDL Generation Using LLM Feedback," *arXiv preprint arXiv:2311.04887*, 2023.
[19] Y. Fu, *et al.*, "GPT4AIGChip: Towards Next-Generation AI Accelerator Design Automation via Large Language Models," in *Proc. ICCAD*, IEEE, 2023.
[20] C. Zhou, *et al.*, "Lima: Less is More for Alignment," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[21] T. Ajayi, *et al.*, "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain," in *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, GOMACTech '19, March 2019.
[22] "ChatEDAv1." https://github.com/wuhy68/ChatEDAv1, 2024.
[23] Y. Ni, *et al.*, "Evaluating the robustness to instructions of large language models," *arXiv preprint arXiv:2308.14306*, 2023.
[24] L. Ouyang, *et al.*, "Training language models to follow instructions with human feedback," *ArXiv*, vol. abs/2203.02155, 2022.
[25] X. Li and J. Li, "Angle-optimized text embeddings," *arXiv preprint arXiv:2309.12871*, 2023.
[26] T. Dettmers, *et al.*, "QLoRa: Efficient Finetuning of Quantized LLMs," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[27] W. Lian, *et al.*, "SlimOrca: An Open Dataset of GPT-4 Augmented FLAN Reasoning Traces, with Verification." https://huggingface.co/datasets/Open-Orca/SlimOrca, 2023.
[28] "Claude." https://www.anthropic.com/claude, 2024.
[29] S. Kulal, *et al.*, "Spoc: search-based pseudocode to code," in *Proc. NeurIPS*, (Red Hook, NY, USA), Curran Associates Inc., 2019.

Appendix

A. Abstract

This paper introduces OpenROAD-Assistant, an open-source large language model for OpenROAD that responds to queries in either prose or Python script using the OpenROAD APIs. This artifact is released on Zenodo and includes two adaptors: one for prose queries and another for OpenROAD API queries. This appendix describes the method for inferring on OpenROAD-related questions using the artifact, OpenROAD-Assistant.

B. Artifact check-list (meta-information)

- **Algorithm:** The algorithm involves fine-tuning the Llama3-8B foundation model using retrieval-augmented fine-tuning (RAFT) for physical design-specific tasks. It includes techniques like quantized low-rank adaptation (QLoRA) for optimization.
- **Compilation:** Python compiler
- **Model:** Llama3-8B with LORA adaptors, which has been fine-tuned for specific tasks in physical design stage.
- **Dataset: The primary datasets used for fine-tuning are from the EDA Corpus, along with additional public datasets for specific training needs such as SlimOrca for instruction SFT.**
- **Run-time environment:** The run-time environment includes Python (version $>=$ 3.8 but $<$ 3.11) with necessary libraries such as Unsloth, Python-bnb, and other dependencies specified in the requirements.
- **Hardware:** Training was performed using NVIDIA GPUs:Four NVIDIA RTX A5500 GPUs for script adapter training, One NVIDIA V100 GPU for question-answer adapter training.
- **Run-time state:** The run-time state includes the configuration parameters such as max-seq-length, dtype, load-in-4bit, and model-specific parameters like lora-alpha and lora-dropout.
- **Execution:** The model executes fine-tuning and inference scripts to perform tasks like question-answering and script generation for physical design automation.
- **Metrics:** The metrics used to evaluate the model's performance include:pass@k (pass@1 and pass@3) for script generation, BERTScore and BARTScore for question-answering tasks.
- **Output:** The output consists of generated scripts for physical design tasks and responses to specific questions related to OpenROAD.
- **Experiments:** Experiments include fine-tuning the model with different datasets and configurations, and evaluating its performance using the specified metrics. Detailed results and ablation studies are provided in the paper.
- **Publicly available?:** Yes, the model, datasets, and scripts are publicly available. The repository and models can be accessed via the provided links.
- **Code licenses (if publicly available)?: BSD3 open-source license.**
- **Data licenses (if publicly available)?:BSD3 open-source license.**
- **Workflow framework used?:** The training and inference workflows are implemented using Python scripts.
- **Archived (provide DOI)?: https://zenodo.org/doi/10.5281/zenodo.13013369**

C. Description

Obligatory

1) How to access

The repository, including all necessary scripts and datasets, can be accessed on GitHub. The detailed instructions for setup and usage are provided in the README file.

2) Hardware dependencies

NVIDIA GPUs are required for efficient training and fine-tuning. Specific configurations are provided for High Performance Clusters (HPC).

3) Software dependencies

Dependencies include Python ($>=$ 3.8 but $<$3.11), Pip, and additional libraries specified in the requirements.txt file.

4) Commercial software dependencies

There are no commercial software dependencies; the model relies entirely on open-source tools and datasets.

5) Data sets

The primary datasets include the EDA Corpus and SlimOrca for instruction SFT. Additional datasets for specific tasks are available in the repository.

6) Models

The model used is Llama3-8B, fine-tuned using the specified datasets and techniques.

D. Installation

Obligatory Detailed installation instructions are provided in the README file of the repository. This includes setting up the Python environment, installing dependencies, and configuring the training scripts.

E. Experiment workflow

Detailed workflows for both training and inference are provided in the README file and the paper. The steps include:

- Cloning the repository.
- Setting up the Python environment.
- Installing dependencies.
- Configuring and running the training scripts.
- Running inference with the trained models.

F. Evaluation and expected results

Obligatory Evaluation results are provided in the paper, demonstrating the performance of the model on both script generation and question-answering tasks. Expected results include high pass@k scores and high precision, recall, and F1 scores for BERTScore and BARTScore.

G. Experiment customization

Users can customize the experiments by modifying the configuration parameters in the training scripts, such as max-seq-length, dtype, and load-in-4bit, as well as model-specific parameters.

H. Notes

Additional notes and considerations for using the model, such as handling datasets and running inference, are provided in the README file and the paper.

I. Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- http://cTuning.org/ae/submission-20201122.html
- https://github.com/ml-eda/artifact-evaluation/