ChipNeMo: Domain-Adapted LLMs for Chip Design

Mingjie Liu [*1] **Teodor-Dumitru Ene** [*1] **Robert Kirby** [*1] **Chris Cheng** [*1] **Nathaniel Pinckney** [*1] **Rongjian Liang** [*1] **Jonah Alben** [1] **Himyanshu Anand** [1] **Sanmitra Banerjee** [1] **Ismet Bayraktaroglu** [1] **Bonita Bhaskaran** [1] **Bryan Catanzaro** [1] **Arjun Chaudhuri** [1] **Sharon Clay** [1] **Bill Dally** [1] **Laura Dang** [1] **Parikshit Deshpande** [1] **Siddhanth Dhodhi** [1] **Sameer Halepete** [1] **Eric Hill** [1] **Jiashang Hu** [1] **Sumit Jain** [1] **Ankit Jindal** [1] **Brucek Khailany** [1] **George Kokai** [1] **Kishor Kunal** [1] **Xiaowei Li** [1] **Charley Lind** [1] **Hao Liu** [1] **Stuart Oberman** [1] **Sujeet Omar** [1] **Ghasem Pasandi** [1] **Sreedhar Pratty** [1] **Jonathan Raiman** [1] **Ambar Sarkar** [1] **Zhengjiang Shao** [1] **Hanfei Sun** [1] **Pratik P Suthar** [1] **Varun Tej** [1] **Walker Turner** [1] **Kaizhe Xu** [1] **Haoxing Ren** [1]

Abstract

ChipNeMo aims to explore the applications of large language models (LLMs) for industrial chip design. Instead of directly deploying off-the-shelf commercial or open-source LLMs, we instead adopt the following domain adaptation techniques: domain-adaptive tokenization, domain-adaptive continued pretraining, model alignment with domain-specific instructions, and domain-adapted retrieval models. We evaluate these methods on three selected LLM applications for chip design: an engineering assistant chatbot, EDA script generation, and bug summarization and analysis. Our evaluations demonstrate that domain-adaptive pretraining of language models, can lead to superior performance in domain related downstream tasks compared to their base LLaMA2 counterparts, without degradations in generic capabilities. In particular, our largest model, ChipNeMo-70B, outperforms the highly capable GPT-4 on two of our use cases, namely engineering assistant chatbot and EDA scripts generation, while exhibiting competitive performance on bug summarization and analysis. These results underscore the potential of domain-specific customization for enhancing the effectiveness of large language models in specialized applications.

1. Introduction

Over the last few decades, Electronic Design Automation (EDA) algorithms and tools have provided huge gains in chip design productivity. Coupled with the exponential increases in transistor densities provided by Moore's law, EDA has enabled the development of feature-rich complex SoC designs with billions of transistors. More recently, researchers have been exploring ways to apply AI to EDA algorithms and the chip design process to further improve chip design productivity (Khailany et al., 2020; Ren & Fojtik, 2021; Roy et al., 2021). However, many time-consuming chip design tasks that involve interfacing with natural languages or programming languages still have not been automated. The latest advancements in commercial (ChatGPT, Bard, etc.) and open-source (Vicuna (Chiang et al., 2023), LLaMA2 (Touvron et al., 2023), etc.) large language models (LLM) provide an unprecedented opportunity to help automate these language-related chip design tasks. Indeed, early academic research (Thakur et al., 2023; Blocklove et al., 2023; He et al., 2023) has explored applications of LLMs for generating Register Transfer Level (RTL) code that can perform simple tasks in small design modules as well as generating scripts for EDA tools.

We believe that LLMs have the potential to help chip design productivity by using generative AI to automate many language-related chip design tasks such as code generation, responses to engineering questions via a natural language interface, analysis and report generation, and bug triage. In this study, we focus on three specific LLM applications: an **engineering assistant chatbot** for GPU ASIC and Architecture design engineers, which understands internal hardware designs and is capable of explaining complex design topics; **EDA scripts generation** for two domain specific tools based on Python and Tcl for VLSI timing analysis tasks specified in English; **bug summarization and analysis** as part of an internal bug and issue tracking system.

Although general-purpose LLMs trained on vast amounts of internet data exhibit remarkable capabilities in generative AI tasks across diverse domains (as demonstrated in (Bubeck et al., 2023)), recent work such as BloombergGPT (Wu et al., 2023) and BioMedLLM (Venigalla et al., 2022) demonstrate that domain-specific LLM models can outperform a general purpose model on domain-specific tasks. In the hardware design domain, (Thakur et al., 2023; Liu et al., 2023) showed that open-source LLMs (CodeGen (Nijkamp et al.,

[*]Equal contribution [1]NVIDIA.

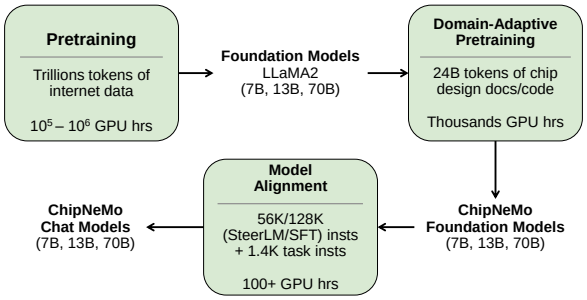

Figure 1: ChipNeMo Training Flow

2023)) fine-tuned on additional Verilog data can outperform state-of-art OpenAI GPT-3.5 models. Customizing LLMs in this manner also avoids security risks associated with sending proprietary chip design data to third party LLMs via APIs. However, it would be prohibitively expensive to train domain-specific models for every domain from scratch, since this often requires millions of GPU training hours. To cost-effectively train domain-specific models, we instead propose to combine the following techniques: Domain-Adaptive Pre-Training (DAPT) (Gururangan et al., 2020) of foundation models with domain-adapted tokenizers, model alignment using general and domain-specific instructions, and retrieval-augmented generation (RAG) (Lewis et al., 2021b) with a trained domain-adapted retrieval model.

As shown in Figure 1, our approach is to start with a base foundational model and apply DAPT followed by model alignment. DAPT, also known as continued pretraining with in-domain data, has been shown to be effective in areas such as biomedical and computer science publications, news, and reviews. In our case, we construct our domain-specific pretraining dataset from a collection of proprietary hardware-related code (e.g. software, RTL, verification testbenches, etc.) and natural language datasets (e.g. hardware specifications, documentation, etc.). We clean up and preprocess the raw dataset, then continued-pretrain a foundation model with the domain-specific data. We call the resulting model a ChipNeMo foundation model. DAPT is done on a fraction of the tokens used in pre-training, and is much cheaper, only requiring roughly 1.5% of the pretraining compute.

LLM tokenizers convert text into sequences of tokens for training and inference. A domain-adapted tokenizer improves the tokenization efficiency by tailoring rules and patterns for domain-specific terms such as keywords commonly found in RTL. For DAPT, we cannot retrain a new domain-specific tokenizer from scratch, since it would make the foundation model invalid. Instead of restricting ChipNeMo to the pre-trained general-purpose tokenizer used by the foundation model, we instead adapt the pre-trained tokenizer to our chip design dataset, only adding new tokens for domain-specific terms.

ChipNeMo foundation models are completion models which require model alignment to adapt to tasks such as chat. We use largely publicly available general-purpose chat instruction datasets for multi-turn chat together with a small amount of domain-specific instruction datasets to perform alignment on the ChipNeMo foundation model, which produces the ChipNeMo chat model. We observe that alignment with a general purpose chat instruction dataset is adequate to align the ChipNeMo foundation models with queries in the chip design domain. We also added a small amount of task-specific instruction data, which further improves the alignment. We trained multiple ChipNeMo foundation and chat models based on variants of LLaMA2 models used as the base foundation model.

To improve performance on the engineering assistant chatbot application, we also leverage Retrieval Augmented Generation (RAG). RAG is an *open-book* approach for giving LLMs precise context for user queries. It retrieves relevant in-domain knowledge from its data store to augment the response generation given a user query. This method shows significant improvement in grounding the model to the context of a particular question. Crucially we observed significant improvements in retrieval hit rate when finetuning a pretrained retrieval model with domain data. This led to even further improvements in model quality.

Our results show that domain-adaptive pretraining was the primary technique driving enhanced performance in domain-specific tasks. We highlight the following contributions and findings for adapting LLMs to the chip design domain:

- We demonstrate domain-adapted LLM effectiveness on three use-cases: an engineering assistant chatbot, EDA tool script generation, and bug summarization and analysis. We achieve a score of 6.0 on a 7 point Likert scale for engineering assistant chatbot based on expert evaluations, more than 70% correctness on the generation of simple EDA scripts, and expert evaluation ratings above 5 on a 7 point scale for summarizations and assignment identification tasks.
- Domain-adapted ChipNeMo models dramatically outperforms all vanilla LLMs evaluated on both multiple-choice domain-specific AutoEval benchmarks and human evaluations for applications.
- Using the SteerLM alignment method (Dong et al., 2023) over traditional SFT improves human evaluation scores for the engineering assistant chatbot by 0.62 points on a 7 point Likert scale.
- SFT on an additional $1.4K$ domain-specific instructions significantly improves the model's proficiency at generating correct EDA tool scripts by 18%.
- Domain-adaptive tokenization reduce domain data token count by up to 3.3% without hurting effectiveness on applications.
- Fine-tuning our ChipNeMo retrieval model with

domain-specific data improves the retriever hit rate by 30% over a pre-trained state-of-the-art retriever, in turn improving overall quality of RAG responses.

The paper is organized as follows. Section 2 outlines domain adaptation and training methods used including the adapted tokenizer, DAPT, model alignment, and RAG. Section 3 describes the experimental results including human evaluations for each application. Section 4 describes relevant LLM methods and other work targeting LLMs for chip design. Finally, detailed results along with additional model training details and examples of text generated by the application use-cases are illustrated in the Appendix.

2. ChipNeMo Domain Adaptation Methods

ChipNeMo implements multiple domain adaptation techniques to adapt LLMs to the chip design domain. These techniques include domain-adaptive tokenization for chip design data, domain adaptive pretraining with large corpus of domain data, model alignment to domain specific tasks, and retrieval-augmented generation with a fine-tuned retrieval model. We illustrate the details of each technique in this section.

2.1. Domain-Adaptive Tokenization

When adapting a pre-trained tokenizer, the main goals are to improve tokenization efficiency on domain-specific data, maintain language model performance on general datasets, and minimize the effort for retraining/fine-tuning. To achieve this, we developed the following approach:

1. Train a tokenizer from scratch using domain-specific data.
2. From the vocabulary of the new tokenizer, identifying tokens that are absent in the general-purpose tokenizer and are rarely found in general-purpose datasets.
3. Expand the general-purpose tokenizer with the newly identified tokens at Step 2.
4. Initialize model embeddings of the new tokens by utilizing the general-purpose tokenizer.

Specifically for Step 4, when a new token is encountered, it is first re-tokenized using the original pretrained general-purpose tokenizer. The LLM's token embedding for the new token is determined by averaging the embeddings of the tokens generated by the general-purpose tokenizer (Koto et al., 2021). The LLM's final output layer weights for the new tokens are initialized to zero.

Step 2 helps maintain the performance of the pre-trained LLM on general datasets by selectively introducing new tokens that are infrequently encountered in general-purpose datasets. Step 4 reduces the effort required for retraining or finetuning the LLM via initialization of the embeddings of new tokens guided by the general-purpose tokenizer.

2.2. Domain Adaptive Pretraining

In our study, we apply DAPT on pretrained foundation base models: LLaMA2 7B/13B/70B. Each DAPT model is initialized using the weights of their corresponding pretrained foundational base models. We name our domain-adapted models **ChipNeMo**. We employ tokenizer augmentation as depicted in Section 2.1 and initialize embedding weight accordingly (Koto et al., 2021). We conduct further pretraining on domain-specific data by employing the standard autoregressive language modeling objective. All model training procedures are conducted using the NVIDIA NeMo framework (Kuchaiev et al., 2019), incorporating techniques such as tensor parallelism (Shoeybi et al., 2019) and flash attention (Dao et al., 2022) for enhanced efficiency.

Our models undergo a consistent training regimen with similar configurations. A small learning rate of $5 \cdot 10^{-6}$ is employed, and training is facilitated using the Adam optimizer, without the use of learning rate schedulers. The global batch size is set at 256, and a context window of 4096 tokens is applied, resulting in an effective batch size of 1M tokens. The total number of training steps is set to 23,200, equating to roughly 1 epoch of the data blend.

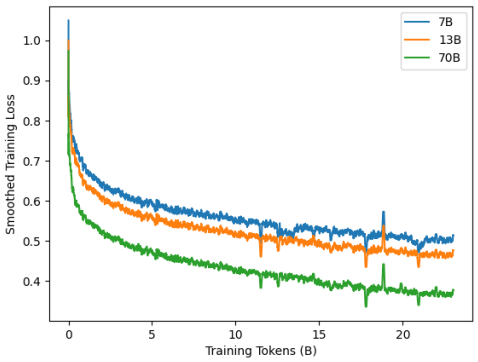

Figure 2: Smoothed Training Loss for ChipNeMo with Tokenizer Augmentation.

Figure 2 illustrates the training loss of ChipNeMo under the specified hyperparameters. We do observe spikes in the training loss. In contrast to the hypothesis in (Chowdhery et al., 2022), we postulate that in our scenario, these spikes can be attributed to "bad data" since these irregularities seem to consistently occur in similar training steps for the same model, even across different model sizes. We chose not to address this issue, as these anomalies did not appear to significantly impede subsequent training steps (with no noticeable degradation in validation loss), possibly due to

our application of a low learning rate.

We refer readers to Appendix for details on the training data collection process A.2, training data blend A.3, and implementation details and ablation studies on domain-adaptive pretraining A.6.

2.3. Model Alignment

After DAPT, we perform model alignment. We specifically leverage two alignment techniques: supervised fine-tuning (SFT) and SteerLM (Dong et al., 2023). We adopt the identical hyperparameter training configuration as DAPT for all models, with the exception of using a reduced global batch size of 128. We employ an autoregressive optimization objective, implementing a strategy where losses associated with tokens originating from the system and user prompts are masked (Touvron et al., 2023). This approach ensures that during backpropagation, our focus is exclusively directed towards the optimization of answer tokens.

We combined our domain alignment dataset, consisting of approximately 1.4k samples, with larger general chat datasets. For SFT, we blended the domain instructional data with 128k commercial-viable chat data and then performed fine-tuning for a single epoch after random shuffling. We conducted experiments involving augmentation of the domain-specific SFT dataset for more than one epoch. However, it became apparent that the model rapidly exhibited signs of overfitting when presented with in-domain questions, often repeating irrelevant answers from the domain SFT dataset. For SteerLM, we closely followed the steps outlined in (Wang et al., 2023). We first trained an attribute model instantiated with LLaMA2-13B model on the HelpSteer and OASST datasets. We then used the attribute model to label all attributes for OASST data and our domain instructional data. Finally, we conducted attribute-conditioned fine-tuning and also masked the attribute labels and trained on ChipNeMo models for 2 epochs. We refer readers to Appendix A.4 for details on the alignment datasets and A.7 on implementations details.

We also experimented with DAPT directly on a chat aligned model, such as the LLaMA2-Chat model. We found that DAPT significantly degraded the model's alignment, making the resulting model useless for downstream tasks.

2.4. Domain-Adapted Retrieval Model

It is well known that LLMs can generate inaccurate text, so-called *hallucination* (Ji et al., 2023). Although the phenomenon is not completely understood, we still must mitigate *hallucinations* since they are particularly problematic in an engineering assistant chatbot context, where accuracy is critical. Our proposal is to leverage the retrieval augmented generation (RAG) method. RAG tries to retrieve relevant passages from a database to be included in the prompt together with the question, which grounds the LLM to produce more accurate answers. We find that using a domain adapted language model for RAG significantly improves answer quality on our domain specific questions. Also, we find that fine-tuning an off-the-shelf unsupervised pre-trained dense retrieval model with a modest amount of domain specific training data significantly improves retrieval accuracy. Our domain-adapted RAG implementation diagram is illustrated on Figure 3.

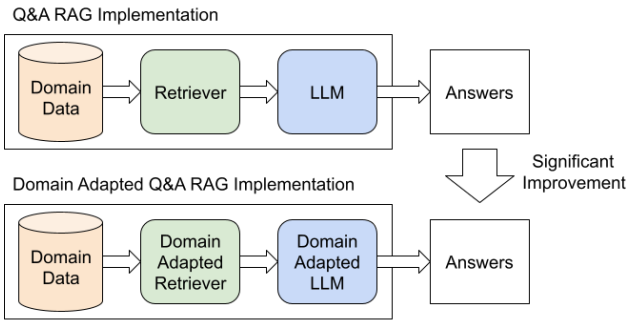

Figure 3: RAG Implementation Variations

We created our domain adapted retrieval model by fine-tuning the *e5_small_unsupervised* model (Wang et al., 2022) with 3000 domain specific auto-generated samples using the Tevatron framework (Gao et al., 2022). We refer readers to the details on the sample generation and training process in Appendix A.8.

Even with the significant gains that come with fine-tuning a retrieval model, the fact remains that retrieval still struggles with queries that do not map directly to passages in the document corpus or require more context not present in the passage. Unfortunately, these queries are also more representative of queries that will be asked by engineers in real situations. Combining retrieval with a domain adapted language model is one way to address this issue.

3. Evaluations

We evaluate our training methodology and application performance in this section. We study our 7B, 13B, and 70B models in the training methodology evaluation, and only our ChipNeMo-70B model using SteerLM for model alignment in the application performance evaluation. For comparison, we also evaluate two baseline chat models: LLaMA2-70B-Chat and GPT-4. LLaMA2-70B-Chat is the publicly released LLaMA2-Chat model trained with RLHF and is considered to be the state-of-the-art open-source chat model, while GPT-4 is considered to be the state-of-the-art proprietary chat model.

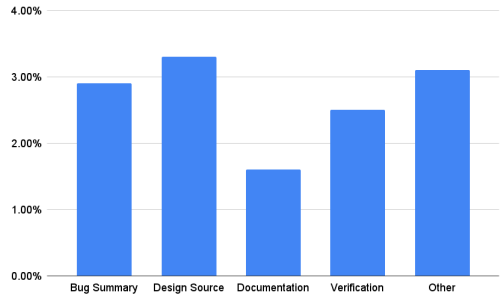

Figure 4: Domain-Adapted ChipNeMo Tokenizer Improvements.

3.1. Domain-Adaptive Tokenization

We adapt the LLaMA2 tokenizer (containing 32K tokens) to chip design datasets using the previously outlined four-step process. Approximately 9K new tokens are added to the LLaMA2 tokenizer. The adapted tokenizers can improve tokenization efficiency by 1.6% to 3.3% across various chip design datasets as shown in Figure 4. We observe no obvious changes to tokenizer efficiency on public data. Importantly, we have not observed significant decline in the LLM's accuracy on public benchmarks when using the domain-adapted tokenizers even prior to DAPT.

3.2. Domain Adaptive Pretraining

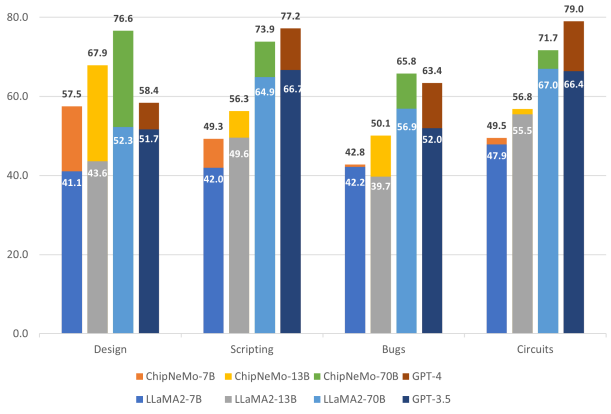

Figure 5: Chip Domain Benchmark Result for ChipNeMo.

Figure 5 presents the outcomes for ChipNeMo models on the AutoEval benchmark for chip design domain (detailed in Appendix A.5). Results on open domain academic benchmark results are presented in Appendix A.6. Our research findings can be summarized as follows:

- DAPT exerts a substantial positive impact on tasks within the domain itself. This effect is manifested in significant improvements in internal design knowledge as well as general circuit design knowledge.
- DAPT models exhibit a slight degradation in performance on open-domain academic benchmarks.
- The use of larger and more performant foundational models yields better zero-shot results on domain-specific tasks. Furthermore, the employment of superior base models results in enhanced domain models post-DAPT, leading to heightened performance on in-domain tasks.
- Improvements attributed to DAPT with in-domain tasks exhibit a positive correlation with model size, with larger models demonstrating more pronounced enhancements in domain-specific task performance.

3.3. Training Ablation Studies

For our ablation studies, we conducted multiple rounds of domain adaptive pre-training. We provide brief summaries and refer to the Appendix A.6 for details.

The differences between training with the augmented tokenizer and the original tokenizer appeared to be negligible. We thus primarily attribute the accuracy degradation on open-domain academic benchmarks to domain data. Moreover, the removal of the public dataset only slightly regressed on most tasks including academic benchmarks.

In our exploration, we experimented with employing a larger learning rate, as in CodeLLaMA (Rozière et al., 2023). We observed large spikes in training loss at the initial training steps. Although this approach eventually led to improved training and validation loss, we noted substantial degradations across all domain-specific and academic benchmarks, except on coding. We hypothesize that a smaller learning rate played a dual role, facilitating the distillation of domain knowledge through DAPT while maintaining a balance that did not veer too far from the base model, thus preserving general natural language capabilities.

We also explored the application of Parameter Efficient Fine-Tuning (PEFT) in the context of Domain-Adaptive Pre-training (DAPT). In this pursuit, we conducted two experiments involving the incorporation of LoRA adapters (Hu et al., 2021), introducing additional parameters of 26.4 million (small) and 211.2 million (large) respectively. In both instances, our findings revealed a significant accuracy gap on in-domain tasks when compared to the full-parameter DAPT approach. Furthermore, when contrasting the outcomes between small and large PEFT models, we observed a marginal enhancement on in-domain task accuracy, with large adapter exhibiting a slight improvement.

3.4. Training Cost

All models have undergone training using 128 A100 GPUs. We estimate the costs associated with domain adaptive pre-training for ChipNeMo as illustrated in Table 1. It is worth noting that DAPT accounts for less than 1.5% of the overall

cost of pretraining a foundational model from scratch.

Model Size	Pretraining	DAPT	SFT
7B	184,320	2,620	90
13B	368,640	4,940	160
70B	1,720,320	20,500	840

Table 1: Training cost of LLaMA2 models in A100 GPU hours. Pretraining cost from (Touvron et al., 2023).

3.5. RAG and Engineering Assistant Chatbot

We created a benchmark to evaluate the performance of design chat assistance, which uses the RAG method. This benchmark includes 88 questions in three categories: architecture/design/verification specifications (Specs), testbench regression documentation (Testbench), and build infrastructure documentation (Build). For each question, we specify the golden answer as well as the paragraphs in the design document that contains the relevant knowledge for the answer. These questions are created by designers manually based on a set of design documents as the data store for retrieval. It includes about 1.8K documents, which were segmented into 67K passages, each about 512 characters.

First, we compare our domain adapted retrieval model with Sentence Transformer (Reimers & Gurevych, 2019) and *e5_small_unsupervised* (Wang et al., 2022) on each category. Each model fetches its top 8 passages from the data store.

As shown in Figure 6, our domain-adapted model performed 2x better than the original *e5_small_unsupervised* model and 30% better than sentence transformer.

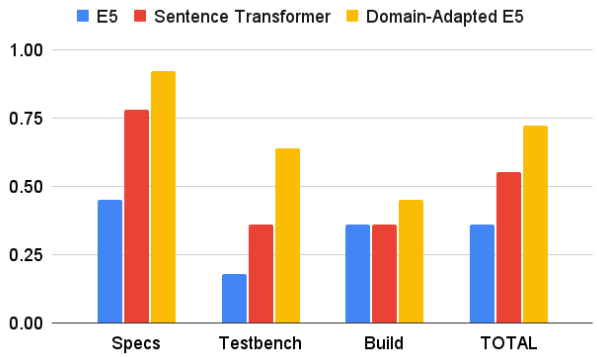

Figure 6: Retrieval Model Accuracy Comparison

The queries in the Specs category are derived directly from passages in the documents, so their answers are often nicely contained in a concise passage and clearly address the query. On the other hand, the queries of the Testbench and Build categories are not directly derived from passages, so their answers were often not as apparent in the fetched passages and required more context (see Appendix A.8 for detailed examples). This significantly contributes to the difference in retrieval quality between the categories.

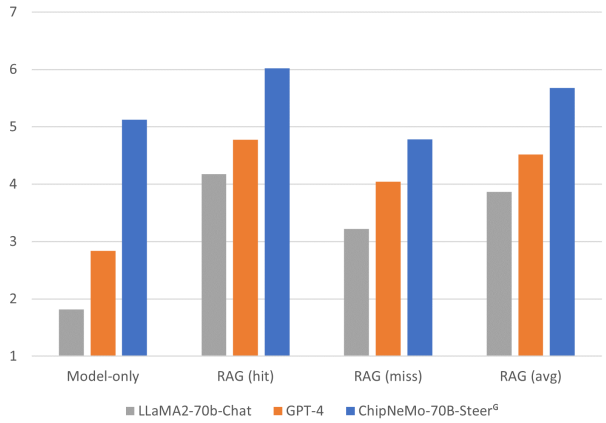

Figure 7: Human Evaluation of Different Models. Model Only represents results without RAG. RAG (hit)/(miss) only include questions whose retrieved passages hit/miss their ideal context, RAG (avg) includes all questions. 7 point Likert scale.

We conducted evaluation of multiple ChipNeMo models and LLaMA2 models with and without RAG. The results were then scored by human evaluators on a 7 point Likert scale and shown in Figure 7. We highlight the following:

- ChipNeMo-70B-Steer outperforms GPT-4 in all categories, including both RAG misses and hits.
- ChipNeMo-70B-Steer outperforms similar sized LLaMA2-70b-Chat in model-only and RAG evaluations by 3.31 and 1.81, respectively.

Our results indicate that RAG significantly boosts human scores. RAG improves ChipNeMo-70B-Steer, GPT-4, and LLaMA2-70b-Chat by 0.56, 1.68, and 2.05, respectively. Even when RAG misses, scores are generally higher than without using retrieval. The inclusion of relevant in-domain context still led to improved performance, as retrieval is not a strictly binary outcome. Furthermore, while ChipNeMo-70B-SFT outperforms GPT4 by a large margin through traditional supervised fine-tuning, applying SteerLM methods (Wang et al., 2023) leads to further elevated chatbot ratings. We refer readers to the complete evaluation results in Appendix A.9.

3.6. EDA Script Generation

In order to evaluate our model on the EDA script generation task, we created two different types of benchmarks. The first is a set of "Easy" and "Medium" difficulty tasks (1-4 line solutions) that can be evaluated without human intervention by comparing with a golden response or comparing the generated output after code execution. The second set of tasks "Hard" come from real use case scenarios that our engineers chose. These tasks are much harder requiring multiple API calls and understanding relationship between

different VLSI objects. Because these are hard to evaluate in an automated way (with current model performance), we had human engineers judge the correctness between 0-10. We evaluate the model on two tools, one is a fully in-house Python based tool and the other is a Tcl based EDA tool with limited public data. The size of these benchmarks are described in Table 2. Work is ongoing to both increase the size and scope for these benchmarks to allow us to further assess and improve these models.

Evaluation Benchmark Name	Size
Python Tool - Automatic (Easy)	146
Python Tool - Automatic (Medium)	28
Python Tool - Human (Hard)	25
Tcl Tool - Automatic (Easy)	708
Tcl Tool - Automatic (Medium)	27
Tcl Tool - Human (Hard)	25

Table 2: EDA Script Generation Evaluation Benchmarks

The comparative performance of our models on these evaluations are shown in Figures 8 and 9. Figure 8 shows the results on automated "easy" and "medium" benchmarks where we check for fully accurate code. For "Hard" benchmarks in Figure 9 we check for partial correctness of the code, which is evaluated by a human user on a 0-10 scale. ChipNeMo-70B-Steer performs significantly better than off-the-shelf GPT-4 and LLaMA2-70B-Chat model.

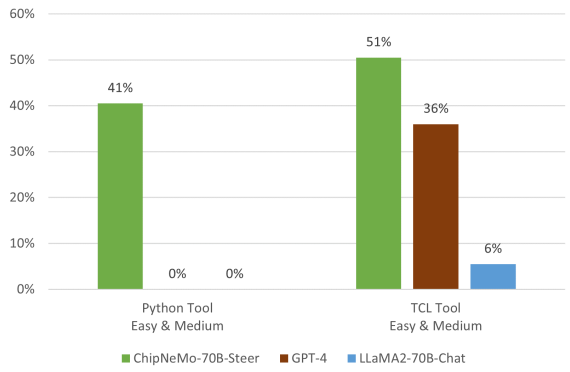

Figure 8: EDA Script Generation Evaluation Results, Pass@5

As seen in Figure 8, models like GPT-4 and LLaMA2-70B-Chat have close to zero accuracy for the Python tool where the domain knowledge related to APIs of the tool are necessary. This shows the importance of DAPT. Without DAPT, the model had little to no understanding of the underlying APIs and performed poorly on both automatic and human evaluated benchmarks. Our aligned model further improved the results of DAPT because our domain instructional data helps guide the model to present the final script in the most useful manner. An ablation study on inclusion of domain instructional data for model alignment and the application of retrieval is provided in Appendix A.9.

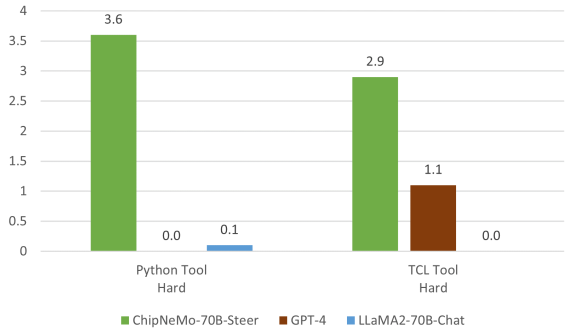

Figure 9: EDA Script Generation Evaluation Results, Single Generation (temperature=0), Human Evaluated 0-10 Point Scale.

Our non-domain models performed better on our Tcl tool than the Python tool, but the trend for our domain trained model was the opposite. We suspect this was due to the proprietary nature of our Python tool. It was difficult for general LLMs to perform well on our Python tool benchmark without knowledge of the APIs. Since ChipNeMo is trained with domain data, the inherent python coding ability of the base model allows ChipNeMo-70B-Steer to perform better. This again highlights the importance of DAPT for low-volume or proprietary programming languages.

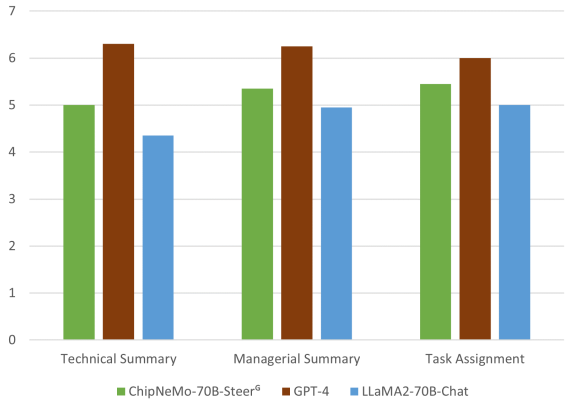

Figure 10: Bug Summarization and Analysis Evaluation Results, 7 point Likert scale.

3.7. Bug Summarization and Analysis

To evaluate our models on bug summarization and analysis we have a hold out set of 30 bugs which are ideal candidates for summarization. This includes having a long comment history or other data which makes the bugs hard for a human to quickly summarize. As described in Appendix A.10.3 the long length of each individual bug requires the LLM to perform hierarchical summarization.

We study three separate sub-tasks: summarization focused on technical details, summarization focused on managerial details, and a post-summarization recommendation of

task assignnment. Participants are tasked with rating the model's performance on a 7-point Likert scale for each of these three assignments. The results can be found in Figure 10. Although the GPT-4 model excels in all three tasks, outperforming both our ChipNeMo-70B-Steer model and the LLaMA2-70B-Chat model, ChipNeMo-70B-Steer does exhibit enhancements compared to the off-the-shelf LLaMA model of equivalent size. We attribute the comparatively lower improvements in summarization tasks resulting from our domain-adaptation to the limited necessity for domain-specific knowledge in summarization compared to other use-cases.

4. Related Works

Many domains have a significant amount of proprietary data which can be used to train a domain-specific LLM. One approach is to train a domain specific foundation model from scratch, e.g., BloombergGPT(Wu et al., 2023) for finance, BioMedLLM(Venigalla et al., 2022) for biomed, and Galactica(Taylor et al., 2022) for science. These models were usually trained on more than 100B tokens of raw domain data. The second approach is domain-adaptive pretraining (DAPT) (Gururangan et al., 2020) which continues to train a pretrained foundation model on additional raw domain data. It shows slight performance boost on domain-specific tasks in domains such as biomedical, computer science publications, news, and reviews. In one example, (Lewkowycz et al., 2022) continued-pretrained a foundation model on technical content datasets and achieved state-of-the-art performance on many quantitative reasoning tasks.

Retrieval Augmented Generation (RAG) helps ground the LLM to generate accurate information and to extract up-to-date information to improve knowledge-intensive NLP tasks (Lewis et al., 2021a). It is observed that smaller models with RAG can outperform larger models without RAG (Borgeaud et al., 2022). Retrieval methods include sparse retrieval methods such as TF-IDF or BM25(Robertson & Zaragoza, 2009), which analyze word statistic information and find matching documents with a high dimensional sparse vector. Dense retrieval methods such as (Karpukhin et al., 2020; Izacard et al., 2022a) find matching documents on an embedding space generated by a retrieval model pretrained on a large corpus with or without fine-tuning on a retrieval dataset. The retrieval model can be trained standalone (Karpukhin et al., 2020; Izacard et al., 2022a; Shi et al., 2023) or jointly with language models (Izacard et al., 2022b; Borgeaud et al., 2022). In addition, it has been shown that off-the-shelf general purpose retrievers can improve a baseline language model significantly without further fine-tuning (Ram et al., 2023). RAG is also proposed to perform code generation tasks (Zhou et al., 2023) by retrieving from coding documents.

Foundation models are completion models, which have limited chat and instruction following capabilities. Therefore, a model alignment process is applied to the foundation models to train a corresponding chat model. Instruction fine-tuning (Wei et al., 2022) and reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022) are two common model alignment techniques. Instruction fine-tuning further trains a foundation model using instructions datasets. RLHF leverages human feedback to label a dataset to train a reward model and applies reinforcement learning to further improve models given the trained reward model. RLHF is usually more complex and resource hungry than instruction fine-tuning. Therefore, recent studies also propose to reduce this overhead with simpler methods such as DPO (Rafailov et al., 2023) and SteerLM (Dong et al., 2023).

Researchers have started to apply LLM to chip design problems. Early works such as Dave (Pearce et al., 2020) first explored the possibility of generating Verilog from English with a language model (GPT-2). Following that work, (Thakur et al., 2023) showed that fine-tuned open-source LLMs (CodeGen) on Verilog datasets collected from GitHub and Verilog textbooks outperformed state-of-the-art OpenAI models such as *code-davinci-002* on 17 Verilog questions. (Liu et al., 2023) proposed a benchmark with more than 150 problems and demonstrated that the Verilog code generation capability of pretrained language models could be improved with supervised fine-tuning by bootstrapping with LLM generated synthetic problem-code pairs. Chip-Chat (Blocklove et al., 2023) experimented with conversational flows to design and verify a 8-bit accumulator-based microprocessor with GPT-4 and GPT-3.5. Their findings showed that although GPT-4 produced relatively high-quality codes, it still does not perform well enough at understanding and fixing the errors. ChipEDA (He et al., 2023) proposed to use LLMs to generate EDA tools scripts. It also demonstrated that fine-tuned LLaMA2 70B model outperforms GPT-4 model on this task.

5. Conclusions

We explored domain-adapted approaches to improve LLM performance for industrial chip design tasks. Our results show that domain-adaptive pretrained models, such as the 7B, 13B, and 70B variants of ChipNeMo, achieve similar or better results than their base LLaMA2 models with only 1.5% additional pretraining compute cost. Our largest trained model, ChipNeMo-70B, also surpasses the much more powerful GPT-4 on two of our use cases, engineering assistant chatbot and EDA scripts generation, while showing competitive performance on bug summarization and analysis. Our future work will focus on further improving ChipNeMo models and methods for production use.