MG-Verilog: Multi-grained Dataset Towards Enhanced LLM-assisted Verilog Generation

Yongan Zhang, Zhongzhi Yu, Yonggan Fu, Cheng Wan, Yingyan (Celine) Lin
celine.lin@gatech.edu
Georgia Institute of Technology
Atlanta, Gerogia, USA

***Abstract*—Large Language Models (LLMs) have recently shown promise in streamlining hardware design processes by encapsulating vast amounts of domain-specific data. In addition, they allow users to interact with the design processes through natural language instructions, thus making hardware design more accessible to developers. However, effectively leveraging LLMs in hardware design necessitates providing domain-specific data during inference (e.g., through in-context learning), fine-tuning, or pre-training. Unfortunately, existing publicly available hardware datasets are often limited in size, complexity, or detail, which hinders the effectiveness of LLMs in hardware design tasks. To address this issue, we first propose a set of criteria for creating high-quality hardware datasets that can effectively enhance LLM-assisted hardware design. Based on these criteria, we propose a Multi-Grained-Verilog (MG-Verilog) dataset, which encompasses descriptions at various levels of detail and corresponding code samples. To benefit the broader hardware design community, we have developed an open-source infrastructure that facilitates easy access, integration, and extension of the dataset to meet specific project needs. Furthermore, to fully exploit the potential of the MG-Verilog dataset, which varies in complexity and detail, we introduce a balanced fine-tuning scheme. This scheme serves as a unique use case to leverage the diverse levels of detail provided by the dataset. Extensive experiments demonstrate that the proposed dataset and fine-tuning scheme consistently improve the performance of LLMs in hardware design tasks.**

I. Introduction

Large Language Models (LLMs) have recently emerged as a promising approach to streamline hardware design processes [1], [4], [5], [8], [16], [17]. By encapsulating vast amounts of domain-specific data and enabling users to interact with the design processes through natural language prompts, LLMs have the potential to make hardware design more accessible to a broader range of developers. This increased accessibility can foster innovation and accelerate the development of new hardware solutions, as it allows developers with varying levels of expertise to contribute to design processes.

Despite the great potential of LLMs, existing state-of-the-art (SOTA) general LLMs, e.g., OpenAI's GPT-4 [11], are still limited in their ability to generate practical hardware designs. For example, they might generate non-synthesizable or non-functional hardware source code [4]. To address this limitation, recent studies suggest that incorporating additional domain-specific data is crucial for enhancing LLMs' performance in hardware design tasks, using techniques across the scopes of LLM inference, fine-tuning, or pre-training. Specifically, one approach to improve LLMs' hardware design capabilities is to provide them with additional relevant design examples during inference-only generation, e.g., GPT4AIGChip [4]. It has been shown that this method can significantly enhance the quality of generated High-Level Synthesis (HLS) hardware code. Another approach is to fine-tune LLMs on carefully curated hardware design datasets, e.g., VerilogEval [8], which has been shown to improve LLMs' performance in generating Verilog code. Alternatively, LLMs can also be pre-trained on diverse datasets from various hardware design domains to specialize in general hardware design concepts, as exemplified by ChipNemo [7], leading to improved general performance across a range of hardware design tasks.

Although the aforementioned approaches show promise in enhancing LLMs' performance in hardware design tasks, their progress can be hindered by the limitations of current publicly available hardware design datasets. As we will later analyze, the size, complexity, and detail granularity of datasets are essential factors for improving LLMs' performance. However, existing datasets often fall short in one or more of these aspects. Some datasets, e.g., those used in [1], [9], [15], contain only a small number of data points (e.g., under 2e2), which are only suitable for benchmarking the LLMs' task performance but is insufficient for effectively fine-tuning LLMs. Other datasets, like those employed in [8], [16], can be simplistic, either lacking important features (e.g., code samples containing multiple module instantiations and aligned descriptions) or providing only high-level descriptions for each code piece. This simplicity can limit the fine-tuned LLMs' generalization performance when faced with diverse user instructions, thus reducing their effectiveness.

To address the limitations of existing datasets and unlock the full potential of LLM fine-tuning and in-context learning for hardware design tasks, we propose a Multi-Grained-Verilog (MG-Verilog) dataset. This dataset includes hardware descriptions at different levels of detail and their corresponding Verilog code samples with varying design complexity. These features make it suitable for both inference and fine-tuning stages of LLMs to enhance their performance in hardware design tasks. Our main contributions can be summarized as follows:

- We introduce a set of essential criteria for high-quality hardware datasets that can be effectively utilized by LLM-assisted hardware design techniques. These criteria can serve as a guide for the development of future datasets in this domain.

979-8-3503-7608-1/24$31.00 ©2024 IEEE

- We present an open-source MG-Verilog dataset[1], which meets the aforementioned criteria. Additionally, we provide the necessary infrastructure for users to access, integrate, and extend the dataset for their specific project needs, promoting collaboration and facilitating further research in this area.
- We demonstrate a unique use case of the MG-Verilog dataset by proposing a balanced fine-tuning scheme that leverages the diverse levels of detail provided by the dataset. This scheme validates and showcases the potential of the dataset to enable novel approaches in LLM-assisted hardware design.
- Extensive experiments show that LLMs fine-tuned with our MG-Verilog dataset outperform those trained on datasets from other sources in terms of both code implementation accuracy and the sophistication of generated hardware designs. These results highlight the effectiveness of our dataset in enhancing LLMs' performance for hardware design tasks.

II. Criteria for Datasets in LLM-assisted Hardware Design

To create a high-quality dataset for LLM-assisted hardware design, we first establish design criteria to guide the development of the MG-Verilog dataset.

Sufficient dataset size. This is crucial for both training (i.e., domain-specific pre-training or fine-tuning) and inference (i.e., in-context learning) of LLMs. A larger dataset provides diverse examples for improved generalization performance during training [7], [8] and enables effective techniques such as Retrieval-Augmented-Generation (RAG) for enhanced generation quality during inference [4].

Accurate code-description pairs. Each code sample needs to be correct, functional, and associated with a precise description of its functionality. Inaccuracies or ambiguity can mislead LLMs during fine-tuning or pre-training and lead to erroneous code generation during inference.

Varied description detail levels. They are necessary to address two challenges. Datasets with only high-level descriptions may not provide sufficient detail for accurate code generation or effective LLM training (i.e., fine-tuning or pre-training), especially for complex designs. Conversely, datasets dominated by detailed descriptions may limit practical utility, as LLMs trained on such datasets might require users to provide elaborated prompts, which can be as labor-intensive as coding from scratch. Hence, an effective dataset should incorporate both high-level and detailed descriptions in a proper balance. In particular, high-level descriptions can facilitate user-friendly LLM interactions, while detailed descriptions are crucial for enabling LLMs to create complex designs, offering in-depth guidance for LLMs during training, or serving as a comprehensive reference during inference.

Extensibility and integrability for future development. A high-quality hardware dataset should be designed with the research community in mind, allowing for easy extension and integration into various projects. The rapidly evolving nature of hardware design necessitates a dataset that can adapt to the latest trends and requirements. Moreover, the vast scope of hardware design means that different developers may have specific focused areas, making it challenging for a single organization to cover all possible scenarios in a one-time effort. To address this issue, the dataset should be structured in a way that encourages researchers to contribute to its growth and adapt it to their specific needs, fostering collaboration within the research community and ensuring its relevance and utility. This approach not only benefits individual projects but also contributes to the overall advancement of LLM-assisted hardware design methodologies.

[1]https://github.com/luke-avionics/mg-verilog

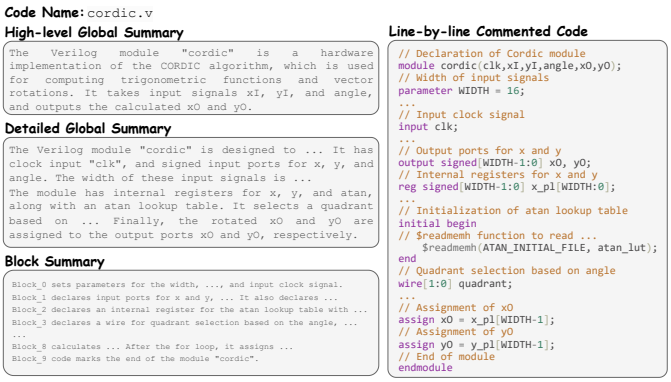


Fig. 1: Illustrating the proposed MG-Verilog dataset structure and examples of varying levels of detail.

III. The Proposed MG-Verilog Dataset

A. Dataset Overview

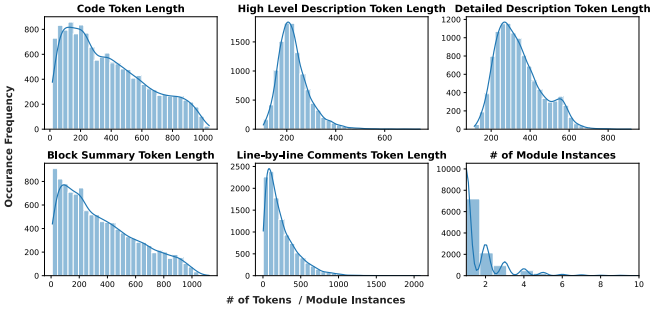
The MG-Verilog dataset consists of over 11,000 Verilog code samples and their corresponding natural language descriptions, serving as the desired outputs and test inputs for various LLM-assisted hardware design tasks, such as Verilog code generation.

B. Dataset Construction

The construction of the MG-Verilog dataset involves several steps to ensure the quality and usability of the data.

1) Data Collection and Preprocessing: Raw source code from open-source repositories is collected and preprocessed to ensure correctness. Adapting from VerilogEval [8], we use Pyverilog [14] to parse the raw Verilog code and exclude code samples containing syntax errors. Deduplication techniques are applied to remove redundant code samples. Additionally, dependencies of the code samples are extracted, i.e., sub-modules of multi-module code samples are identified and recorded as metadata to facilitate research on techniques such as few-shot learning and RAG for generating multi-module Verilog code.

2) Description Generation: Natural language descriptions are appended to the code samples using an approach similar to VerilogEval [8], leveraging LLMs' superior natural language generation capabilities. In addition to simple high-level



Total # of Code Tokens	Total # of High Level Descriptions Tokens	Total # of Detailed Description Tokens	Total # of Block Summary Tokens	Total # of Line-by-line Comments Tokens	Single Module Percentage	Multi-module Percentage
5.3e6	2.6e6	3.8e6	3.9e6	2.7e6	64%	36%

Fig. 2: The detailed statistics of the MG-Verilog dataset, using the tokenizer from the GPT-3.5-Turbo model [10].

descriptions for each code piece, varying levels of detailed descriptions aligned with the code complexity are provided, as detailed in Sec. III-C.

C. Multi-grained Dataset Structure

To strike a balance between design generation accuracy and user-friendliness, we adopt a multi-grained data structure, which encompasses descriptions at various levels of detail in order to satisfy the third criterion in Sec. II. As depicted in Fig. 1, this structure organizes hardware code descriptions, ranging from high-level summaries to detailed, line-by-line comments. The multi-grained structure is designed to mimic the learning and design processes of human designers. The objective is to simplify the learning curve for using the dataset and, as demonstrated later, to better leverage the strengths of LLMs for enhanced description generation accuracy. Specifically, the multi-grained structure mirrors the typical two phases experienced by human designers. In the learning phase, a hardware designer starts with the basic syntax and semantics of the design language, gradually advancing to apply this knowledge to design higher-level hardware modules. Conversely, in the design phase, the process begins with high-level architectural planning for the entire design, followed by a detailed, step-by-step implementation.

D. Detailed Statistics of the Dataset

Fig. 2 presents detailed statistics of the MG-Verilog dataset, illustrating the distribution of token length for both the code and varying levels of descriptions. The complexity of the code samples is also reflected in the distribution of the number of module instances. The dataset shows a wide range of natural language description details and code complexities, making it suitable for diverse LLM-assisted hardware design tasks.

E. Dataset Access and Extension Instructions

The MG-Verilog dataset is publicly available and packaged in the standard HuggingFace Datasets format [6] for easy access and integration. Each dataset entry contains the following fields: *code*, *high-level summaries*, *detailed summaries*, *block-level summaries*, *line-by-line comments*, and *metadata*. The metadata field currently includes the module dependencies of the code samples. The MG-Verilog dataset is open-sourced from raw data collection to the final dataset construction in a modular manner for straightforward extension. The demonstrated balanced fine-tuning use case is also provided as a reference.

IV. Dataset Unique Use Case: A Balanced Fine-tuning Scheme

In this section, we show a unique use case of our proposed MG-Verilog dataset. Specifically, we introduce a balanced fine-tuning scheme to fully harness the diverse levels of detail provided by our MG-Verilog dataset.

The challenge to address. The ultimate goal of fine-tuning is to generate hardware code solely from high-level design descriptions. However, challenges arise when determining the type of descriptions to be used for fine-tuning. On the one hand, fine-tuning with only simple high-level descriptions may not provide LLMs with sufficient information to generate code for complex designs. On the other hand, exclusively relying on detailed descriptions could hinder LLMs' ability to respond to more high-level user instructions.

Our balanced fine-tuning scheme. To tackle the aforementioned challenge, we present a balanced fine-tuning scheme that randomly selects training samples with varying levels of descriptions from the MG-Verilog dataset in each fine-tuning iteration. The aim is to achieve a balance when imparting knowledge of both global and local code semantics to LLMs.

V. Experimental Results

A. Experiment Setup

Dataset generation. The primary model for generating descriptions is LLaMA2-70B-Chat. GPT-3.5-turbo serves as an automated backup for scenarios where the maximum token limit is exceeded. Based on empirical testing, we set the temperature to 0.7 and top_p to 0.95, maintaining other hyperparameters at their default values for the best quality.

Fine-tuning and inference. CodeLLaMA-7B-Instruct is chosen as the primary model for hardware code generation due to its superior coding performance and small model size. For fine-tuning it on our dataset, the fine-tuning approach is based on QLoRA [3], using its default training settings to demonstrate our delivered dataset's effectiveness. The fine-tuned model is evaluated using 143 Verilog coding questions from the benchmark in [8], excluded from the training set.

Hardware evaluation and metrics. The validity of each generated design is tested by compiling it and checking against its RTL simulation results in pre-defined testbench cases. We employ unbiased pass@1, pass@5, and pass@10 metrics, calculated from 20 generation runs, as established in [8].

B. Ablation Study on Different Evaluation Settings

In this section, we explore the performance of fine-tuned models using varying data formats in both the training and evaluation phases. Although high-level global summaries are the most user-friendly data format, their ambiguity often results in a lack of detailed information necessary for precise

TABLE I: Comparison across fine-tuning and evaluation data formats using the CodeLLaMA-7B-Instruct model. The table columns indicate the data formats used for fine-tuning, while the rows show the formats used during evaluation. Performance is color-coded for clarity: warm colors (red and orange) indicate high performance, while cool colors (light blue and blue) denote lower performance. The color gradient from best to worst performance is as follows: **red (highest)**, **orange**, **light blue**, and **blue (lowest)**. A notation of *H*, *MH*, *ML*, and *L* is used to indicate high, medium (high/low), and low performance, respectively, for better visual clarity.

Pass@1

Evaluate \ Fine-tune	MG-Verilog Balanced Fine-tune	High Level Global Summaries	Detailed Global Summaries	Block Summaries
High Level Global Summaries	45.2 *H*	42.4 *ML*	44.8 *MH*	40.3 *L*
Detailed Global Summaries	52.7 *MH*	50.8 *ML*	54.5 *H*	46.3 *L*
Block Summaries	51.1 *MH*	41.8 *ML*	40.0 *L*	52 *H*

Pass@5

Evaluate \ Fine-tune	MG-Verilog Balanced Fine-tune	High Level Global Summaries	Detailed Global Summaries	Block Summaries
High Level Global Summaries	52.2 *H*	48.1 *ML*	49.9 *MH*	44.0 *L*
Detailed Global Summaries	58.5 *MH*	58.5 *MH*	59.7 *H*	52.2 *L*
Block Summaries	56.2 *MH*	52.5 *ML*	46.3 *L*	60 *H*

Pass@10

Evaluate \ Fine-tune	MG-Verilog Balanced Fine-tune	High Level Global Summaries	Detailed Global Summaries	Block Summaries
High Level Global Summaries	55.2 *H*	49.7 *ML*	51.8 *MH*	45.6 *L*
Detailed Global Summaries	60.9 *MH*	61.5 *H*	60.1 *ML*	53.1 *L*
Block Summaries	58.0 *MH*	54.5 *ML*	47.6 *L*	63 *H*

code generation. In some cases, detailed global summaries can actually be more advantageous for expert users who have a deep understanding of code structures. Consequently, an ideal RTL code generation dataset would facilitate consistent model performance across a range of input instruction complexities.

Observations and analysis. Tab. I provides insights into these findings. Notably, we can observe: (1) Models fine-tuned with the MG-Verilog dataset exhibit the most robust performance in all tested evaluation settings. Specifically, while different evaluation settings tend to bias the fine-tuning setting that aligns with them, models fine-tuned with the MG-Verilog dataset consistently rank in the top two positions when compared to other baselines. In contrast, other baselines may perform well only under their aligned evaluation settings and notably under-perform in other evaluation settings. (2) Training exclusively with either overly detailed or overly high-level data can result in decreased performance, indicating the importance of having balanced training data. Specifically, Tab. I reveals that, apart from the MG-Verilog dataset, models trained with detailed global summaries yield the highest pass rates. These summaries strike a balance between the generality of high-level global summaries and the specificity of block summaries.

C. Ablations on the Number of Training Samples

We further examine how the quantity of training samples affects the performance of models fine-tuned for RTL code generation tasks. As illustrated in Fig. 3, there is a clear trend

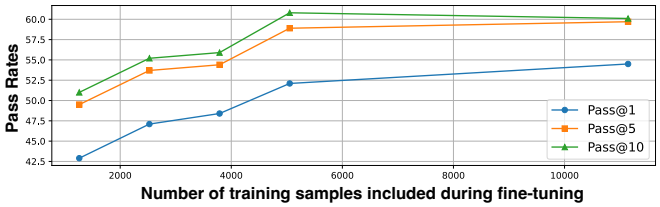


Fig. 3: Pass rates of the generated RTL code from fine-tuned CodeLLaMA-7B-Instruct model using different numbers of training samples. Here only detailed global summaries of the code are used during the fine-tuning.

where the model's performance improves with an increase in the number of training samples. However, we also note a diminishing returns phenomenon. Specifically, the performance gains from additional training samples decrease as the total number of samples grows. This trend could be attributed to either the limited diversity in the raw source code or the potential need for more optimal hyperparameter tuning and model configurations. These aspects, being orthogonal to the dataset structure proposed, are left for future exploration.

VI. Related Work

LLMs have been applied in various stages of the hardware design process, including verification [13], security flaw detection [12], and code generation [2], [4], [8], [16]. However, their performance is still limited due to insufficient exposure to hardware data during pretraining [2], [4]. Some studies [8], [9], [16] have tried to rectify this by supplying more hardware code samples and fine-tuning the LLMs. Yet, the datasets used are still either too small [9] or overly simplistic [8], [16], which hinder effective fine-tuning of LLMs. Our MG-Verilog dataset addresses this issue by providing an open-sourced, high-quality dataset, essential for optimizing LLM fine-tuning and in-context learning.

VII. Conclusion

In this work, we aim to mitigate the limitations of existing datasets for LLM-assisted hardware design by proposing the open-sourced Multi-Grained-Verilog (MG-Verilog) dataset. The MG-Verilog dataset features hardware descriptions at different levels of detail and their corresponding Verilog code samples for more generic use cases. We have demonstrated the effectiveness of the dataset through a balanced fine-tuning scheme. Extensive experiments show that LLMs fine-tuned with the MG-Verilog dataset outperform those trained on other datasets in terms of Verilog code generation accuracy.

VIII. Acknowledgments

The work is supported by the National Science Foundation (NSF) through the RTML funding (Award number: 2400511), an NSF CAREER award (Award number: 2345577), and CoCoSys, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

REFERENCES

[1] J. Blocklove *et al.*, "Chip-chat: Challenges and opportunities in conversational hardware design," *arXiv preprint arXiv:2305.13243*, 2023.

[2] K. Chang *et al.*, "Chipgpt: How far are we from natural language hardware design," *arXiv preprint arXiv:2305.14019*, 2023.

[3] T. Dettmers *et al.*, "Qlora: Efficient finetuning of quantized llms," *arXiv*, 2023.

[4] Y. Fu *et al.*, "Gpt4aigchip: Towards next-generation ai accelerator design automation via large language models," *arXiv preprint arXiv:2309.10730*, 2023.

[5] Z. He *et al.*, "Chateda: A large language model powered autonomous agent for eda," *arXiv preprint arXiv:2308.10204*, 2023.

[6] huggingface, "Datasets," https://huggingface.co/docs/datasets/en/index, (Accessed on 04/01/2024).

[7] M. Liu *et al.*, "Chipnemo: Domain-adapted llms for chip design," *arXiv preprint arXiv:2311.00176*, 2023.

[8] M. Liu *et al.*, "Verilogeval: Evaluating large language models for verilog code generation," *arXiv preprint arXiv:2309.07544*, 2023.

[9] Y. Lu *et al.*, "Rtllm: An open-source benchmark for design rtl generation with large language model," *arXiv preprint arXiv:2308.05345*, 2023.

[10] OpenAI, "Gpt-3.5," https://platform.openai.com/docs/models/gpt-3-5, (Accessed on 04/10/2023).

[11] OpenAI, "Gpt-4 technical report," 2023.

[12] S. Paria *et al.*, "Divas: An llm-based end-to-end framework for soc security analysis and policy-based protection," *arXiv preprint arXiv:2308.06932*, 2023.

[13] P. Srikumar, "Fast and wrong: The case for formally specifying hardware with llms," *ASPLOS Workshop*, 2023.

[14] S. Takamaeda-Yamazaki, "Pyverilog: A python-based hardware design processing toolkit for verilog hdl," in *Applied Reconfigurable Computing*, ser. Lecture Notes in Computer Science, vol. 9040. Springer International Publishing, Apr 2015, pp. 451–460. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-16214-0_42

[15] S. Thakur *et al.*, "Benchmarking large language models for automated verilog rtl code generation," *arXiv preprint arXiv:2212.11140*, 2022.

[16] S. Thakur *et al.*, "Verigen: A large language model for verilog code generation," *arXiv preprint arXiv:2308.00708*, 2023.

[17] Z. Yan *et al.*, "On the viability of using llms for sw/hw co-design: An example in designing cim dnn accelerators," *arXiv preprint arXiv:2306.06923*, 2023.