Towards Benchmarking LLM-based QA for Chip Design

Raymond Baartmans
School of EECS
Oregon State University
Corvallis, OR 97333
baartmar@oregonstate.edu

1 Introduction

Hardware Description Languages (HDLs) such as SystemVerilog are used by hardware engineers to specify the register-transfer level (RTL) structure electronic circuits. RTL design and verification is a crucial phase in the chip design process, as bugs that go unnoticed during this phase can become extremely costly when discovered after first silicon. HDLs are much different from typical imperative programming languages in that consecutive statements in HDL code are not necessarily executed sequentially during RTL simulation. This, combined with the fact that HDLs share a similar syntax to common programming languages like C, can make it difficult for non-experts to correctly and effectively read and comprehend HDL code. This only becomes more time-consuming as the length and complexity of the code increases. Large language models (LLMs) have demonstrated impressive capabilities for generating SystemVerilog code [5, 9, 10, 8], but their ability to perform question-answering on SystemVerilog designs remains unexplored, primarily due to a lack of labeled data for the task. To fill this need, we introduce the `SV-QA` dataset, consisting of 80 question-answer pairs on a variety of different hardware designs expressed in SystemVerilog code. The contributions of this work can be summarized as the following:

1. We introduce a framework for collecting and labeling a SystemVerilog question-answering dataset.
2. We curate and publish `SV-QA`, the first question-answering dataset for SystemVerilog code.
3. We evaluate four popular large language models on the dataset, and compare their performance.

2 Related Work

2.1 Question Answering with LLMs

Question-answering (QA) systems aim to automatically answer questions posed in natural language. This can be useful for many information retrieval and natural language processing tasks. SQuAD [7] is a classic QA dataset for reading comprehension, and has been used as a baseline for evaluating LLMs such as BERT [3]. SQuAD 2.0 [6] is a newer dataset which includes adversarial, unanswerable questions. Other popular QA datasets include HotpotQA [11] for multi-hop reasoning and TriviaQA [2] for trivia questions.

2.2 Adapting LLMs for RTL Design

SystemVerilog [1] is an industry standard programming langauge used for writing RTL descriptions of hardware designs. Many previous works explore adapting LLMs for Verilog code generation [5, 9, 10, 8]. VerilogEval [5] is a dataset for evaluating LLM-generated RTL designs. VeriGen

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

[8] uses public SystemVerilog projects on GitHub to build a large-scale corpus for training LLMs on Verilog code generation. Some works have explored evaluating LLM capabilities on general coding-related QA [4], but none of these works specifically explore QA on SystemVerilog code. Moreover, no datasets or benchmarks exist to tune and evaluate LLM performance on this specific task.

3 Proposed Method

To evaluate LLM performance on QA for SystemVerilog hardware descriptions, we propose `SV-QA`, a QA dataset based on public, freely licensed SystemVerilog code.

3.1 Dataset Design

`SV-QA` consists of 20 SystemVerilog code excerpts, which were manually selected from popular GitHub projects with a primary language of SystemVerilog. All code excerpts in the dataset are freely available under open licenses. For each passage, four hand-written questions were developed, one of which is unanswerable, following a similar methodology to SQuAD 2.0 [6].

3.2 Dataset Makeup

The questions aim to cover syntactical constructs as specified in the IEEE SystemVerilog Standard [1] and common concepts in RTL design, including:

- **Digital Design:** Behavioral and timing analysis of sequential, combinational, and state machine logic.
- **Digital Verification:** Behavioral analysis of functional testbenches, formal properties, and the aspects of the design they are verifying.

Code excerpts and questions were tagged to measure the diversity and coverage of the dataset. These statistics can be viewed in Section A.1.

4 Evaluation

4.1 Experimental Setup

We evaluate four popular open-source LLMs on the `SV-QA` test set: Llama-3-8B, Flan-T5-XXL, Flan-T5-XL, and Mistral-7B. The model implementations were taken from HuggingFace, and the experiments were ran using 3x NVIDIA Tesla V100 GPU. To evaluate model performance, we calculate F1 Score and Exact Match (EM), two commonly used metrics for QA. The source code for the experiments is published at https://github.com/baartmar/LLM-QA-SystemVerilog.

4.2 Results

Model	F1	EM	Latency (s)	F1 (w/Ans)	EM (w/Ans)	F1 (No Ans)	EM (No Ans)
Llama-3-8B	43.34	32.5	3.54	57.79	43.33	0	0
T5-XXL	29.31	25	5.68	24.08	18.33	45	45
Mistral-7B	27.76	20	4.65	37.01	26.67	0	0
T5-XL	24.17	21.25	1.22	7.22	3.33	75	75

Table 1: F1 and EM scores across all questions in the `SV-QA` test set. Avg Latency is average response time in seconds per question. Scores on non-adversarial and adversarial questions are denoted by (w/Ans) and (No Ans) respectively.

Table 1 shows the results from testing various pretrained models on the `SV-QA` test set. Llama-3-8B shows the highest performance, likely due to its more robust pretraining compared to the other models. In the case of Flan-T5 models, increasing parameters raised model performance, as T5-XXL saw a 15 and 17 point increase over T5-XL for F1 and EM score. Notably, decoder-only models struggled with

adversarial questions, as Llama-3-8B and Mistral-7B both did get a single unanswerable question correct.

5 Conclusion

We introduce `SV-QA`, the first question-answering dataset for SystemVerilog code, and evaluate the performance of various popular LLMs on the dataset. Currently, only a test set is available, and is limited in size to 20 SystemVerilog code excerpts and 80 questions. Additionally, due to resource and time constraints, only four models are evaluated, limiting the conclusions which can be drawn from results. Expanding the dataset size, introducing a training corpus, and benchmarking new models remain important directions for future work.

References

[1] Ieee standard for systemverilog–unified hardware design, specification, and verification language. *IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012)*, pages 1–1315, 2018.

[2] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[3] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

[4] Linyi Li, Shijie Geng, Zhenwen Li, Yibo He, Hao Yu, Ziyue Hua, Guanghan Ning, Siwei Wang, Tao Xie, and Hongxia Yang. Inficoder-eval: Systematically evaluating the question-answering capabilities of code large language models, 2024.

[5] Mingjie Liu, Nathaniel Pinckney, Brucek Khailany, and Haoxing Ren. Verilogeval: Evaluating large language models for verilog code generation. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2023.

[6] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[8] Shailja Thakur, Baleegh Ahmad, Zhenxing Fan, Hammond Pearce, Benjamin Tan, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. Benchmarking large language models for automated verilog rtl code generation. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.

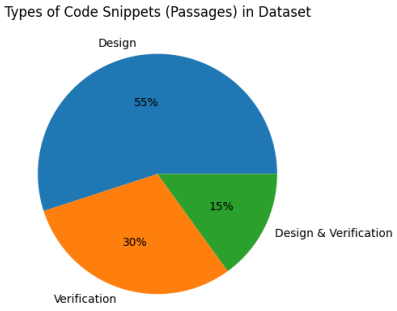
[9] Shailja Thakur, Baleegh Ahmad, Hammond Pearce, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri, and Siddharth Garg. Verigen: A large language model for verilog code generation. *ACM Transactions on Design Automation of Electronic Systems*, 2023.

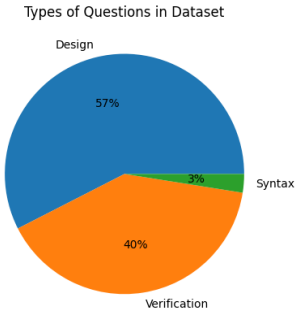
[10] Yun-Da Tsai, Mingjie Liu, and Haoxing Ren. Rtlfixer: Automatically fixing rtl syntax errors with large language models, 2024.

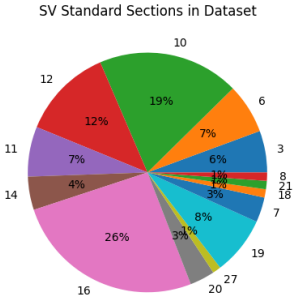
[11] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

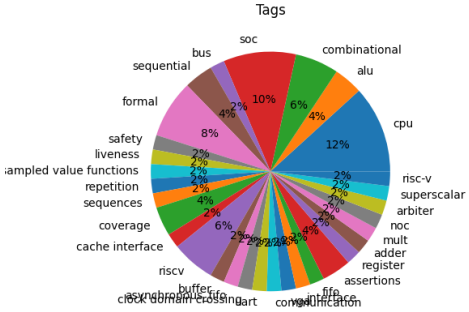
A Appendix

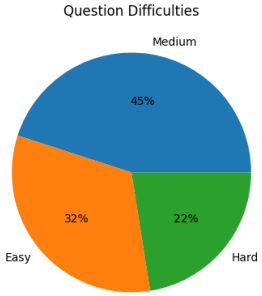
A.1 Dataset Diversity

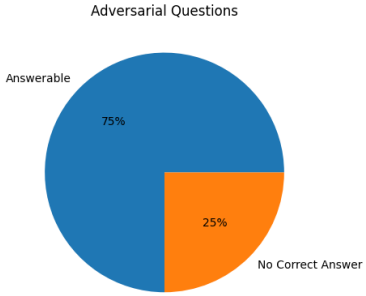

Types of Code Snippets (Passages) in Dataset
Design
55%
15%
Design & Verification
30%
Verification

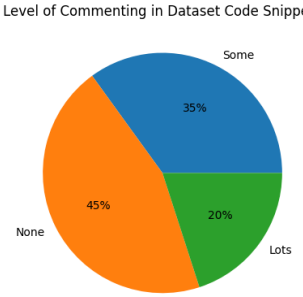

Types of Questions in Dataset
Design
57%
3%
Syntax
40%
Verification


SV Standard Sections in Dataset
10
19%
6
7%
3
6%
8
1%
21
1%
18
3%
7
8%
19
1%
27
3%
20
26%
16
4%
14
7%
11
12%
12


Tags
soc
10%
combinational
6%
alu
4%
cpu
12%
risc-v
2%
superscalar
2%
arbiter
2%
noc
2%
mult
2%
adder
2%
register
2%
assertions
2%
fifo
2%
interface
2%
verification
2%
dart
2%
clock domain crossing
2%
asynchronous fifo
2%
buffer
2%
riscv
2%
cache interface
6%
coverage
2%
sequences
4%
repetition
2%
sampled value functions
2%
liveness
2%
safety
2%
formal
8%
sequential
4%
bus
2%


Question Difficulties
Medium
45%
32%
22%
Easy
Hard


Adversarial Questions
Answerable
75%
25%
No Correct Answer


Level of Commenting in Dataset Code Snippets
Some
35%
45%
20%
None
Lots