



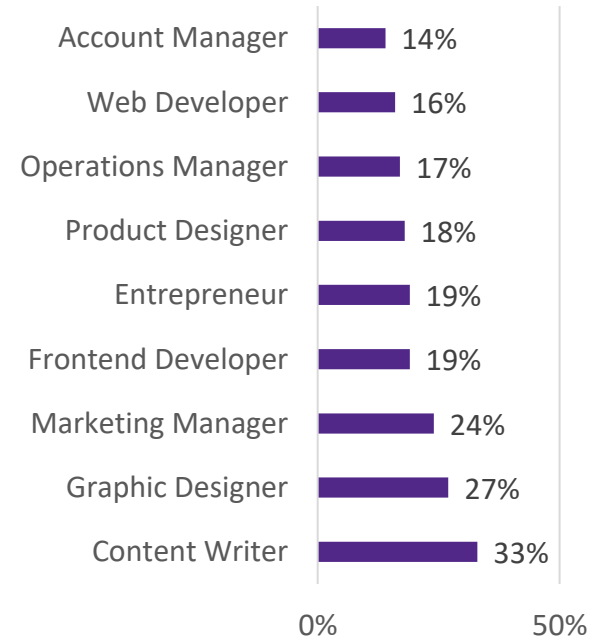
Constructing Secure Hardware Dataset to Support the Application of Large Language Models in Hardware Domain

Weimin Fu and Xiaolong Guo

Kansas State University

Unverified LLMs: A Persistent Security Challenge

- Unverified LLMs will be used, regardless of permission.
- Beyond the obvious risks to **cybersecurity** and **data privacy**, unverified LLMs may **hallucinate** and **systematical errors** in critical tasks.
- Analyzing and exploiting errors produced by LLMs will be a significant security challenge for the foreseeable future.



■ Occupations with the greatest percentage of members on LinkedIn adding AI Aptitude skills to their profiles in 2023

GenAI in the hardware domain

Industry:

Synopsys Announces Synopsys.ai Copilot, Breakthrough GenAI Capability to Accelerate Chip Design

Strategic Collaboration Integrating Microsoft Azure OpenAI Service to Bring the Power of GenAI to Semiconductor Engineering



Research Research Labs ▾ Publications AI Playground ▾ Research Areas ▾ Careers ▾ Licensing

Publications > ChipNeMo: Domain-Adapted LLMs for Chip Design

ChipNeMo: Domain-Adapted LLMs for Chip Design



All Thought Leadership > Verification Horizons

THOUGHT LEADERSHIP

Decoding LLM Hallucinations: Insights and Taming them for EDA Applications

June 15, 2023 • 6 MIN READ



AI/ML for Functional Verification

Academia:

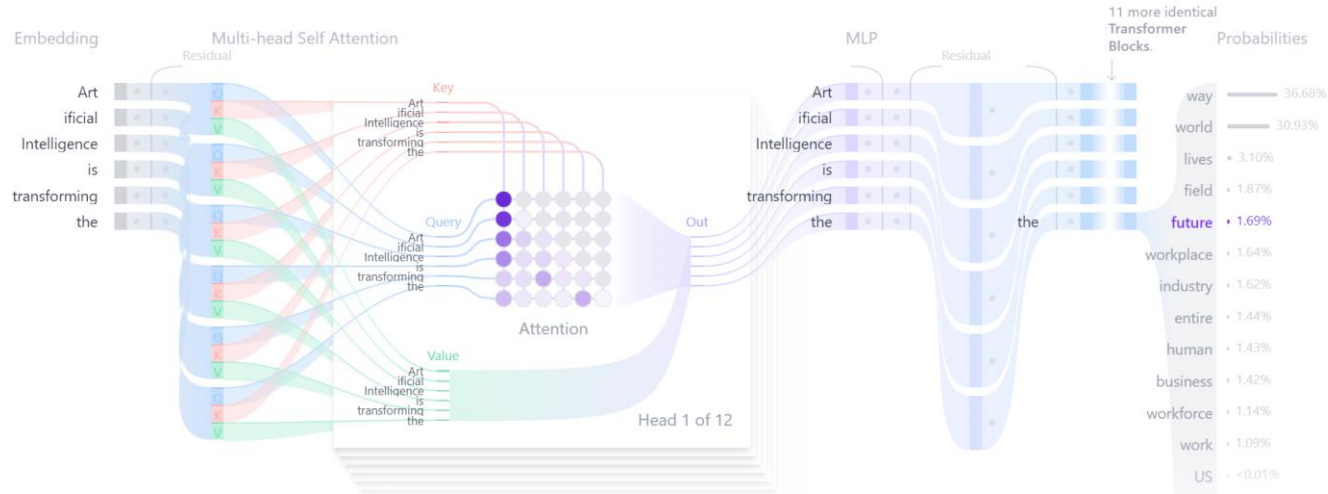
1. Kande, Rahul, et al. "Llm-assisted generation of hardware assertions." *arXiv preprint arXiv:2306.14027* (2023).
2. Ahmad, Baleegh, et al. "Fixing hardware security bugs with large language models." *arXiv preprint arXiv:2302.01215* (2023).
3. Zhang, Zixi, et al. "Llm4dv: Using large language models for hardware test stimuli generation." *arXiv preprint arXiv:2310.04535* (2023).
4. Fang, Wenji, et al. "Assertllm: Generating and evaluating hardware verification assertions from design specifications via multi-llms." *arXiv preprint arXiv:2402.00386* (2024).
5. Saha, Dipayan, et al. "Empowering Hardware Security with LLM: The Development of a Vulnerable Hardware Database." 2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 2024.
6. Hong, Charles, et al. "LLM-Aided Compilation for Tensor Accelerators." *arXiv preprint arXiv:2408.03408* (2024).
7. Ma, Ruiyang, et al. "VerilogReader: LLM-Aided Hardware Test Generation." *arXiv preprint arXiv:2406.04373* (2024).
8. Collini, Luca, Siddharth Garg, and Ramesh Karri. "C2HLSC: Can LLMs Bridge the Software-to-Hardware Design Gap?." *arXiv preprint arXiv:2406.09233* (2024).
9. Xiong, Chenwei, et al. "HLSPilot: LLM-based High-Level Synthesis." *arXiv preprint arXiv:2408.06810* (2024).
10. Goh, Emil, et al. "From English to ASIC: Hardware Implementation with Large Language Model." *arXiv preprint arXiv:2403.07039* (2024).
11.



Let's Challenge the Prompt-Based Approach first

How does LLM work? Predict next token!

Take an example: Artificial Intelligence is transforming the _____



Source: Cho, Aeree, et al. "Transformer Explainer: Interactive Learning of Text-Generative Models." *arXiv preprint arXiv:2408.04619* (2024).



Prompt based Methods

- Incorporating Additional Information in Prompts:
 - Introducing extra information in the prompt can partially address certain issues.
- Limitations of Prompt-Based Methods:
 1. Error in the Prompt:
 - Even minor mistakes in the prompt can drastically alter the output.

ChatGPT4o

System Prompt: You are a helpful assistant

System Prompt: You are a useless assistant

What is the bug inside the design?

Example LED Verilog code from PyVerilog Github Repo

The given design has an issue related to the reset behavior and potential state transition inaccuracies. Here are the points to consider:

1. Assigning the `led` Output:

- The `led` output is assigned the value `count[23:16]`. This is a valid assignment unless you have different intentions for the `led` output based on `count`.

2. Reset Issue:

- The design resets `count` and `state` to zero upon `RST`, which is appropriate and standard.

3. State Transitions:

- The state machine transitions seem logical, but there's an implicit assumption that `enable` is continuously high for correct transitions from `state == 0` to `state == 1`. If `enable` falls before `state` returns to 0, the state machine will remain at `state == 0`.

-

The provided design has a potential bug in how the count and state are reset and updated. Specifically, the bug lies in the way the `count` signal is updated only when the `state` is 2, and there is no mechanism to handle the initial reset condition correctly or ensure that the count operation correctly captures the enable signal condition.



Prompt based Methods

- **Limitations of Prompt-Based Methods:**
 - 1. Error in the Prompt:**
 - Even minor mistakes in the prompt can drastically alter the output.
 - 2. Omission of Key Information in the Prompt:**
 - A lack of critical details can lead to incorrect responses.



ChatGPT4o

Prompt: Could you check CWE-1221? It is about Incorrect Register Defaults.

Prompt: Could you check CWE-1221: Incorrect Register Defaults or Module Parameters for the following code?

CWE-1221 is about hardware description language code incorrectly defining register defaults or hardware Intellectual Property (IP) parameters to insecure values.

Example LED Verilog code from PyVerilog Github Repo

Upon checking the provided Verilog code for Common Weakness Enumeration (CWE) entry CWE-1221, which focuses on a "**Missing Combinational Logic Delay Flop Reset**," I don't see a specific issue related to this CWE in your design.

Reviewing the provided Verilog HDL code in the context of CWE-1221, we should inspect the initial values and conditions of the registers and parameters to ensure there are no insecure defaults or configurations.

Here are some aspects to consider in the code for CWE-1221 compliance:

1. Check Initial Values of Registers:

- Registers `count` and `state` are initialized to zero in the RST block, which typically represents the reset condition.
- Ensure that `0` is a safe default value for both `count` and `state` in the context of your hardware's security requirements.



Prompt based Methods

Limitations of Prompt-Based Methods:

1. Error in the Prompt:

Even minor mistakes in the prompt can drastically alter the output.

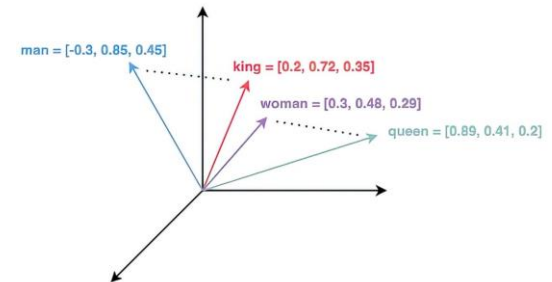
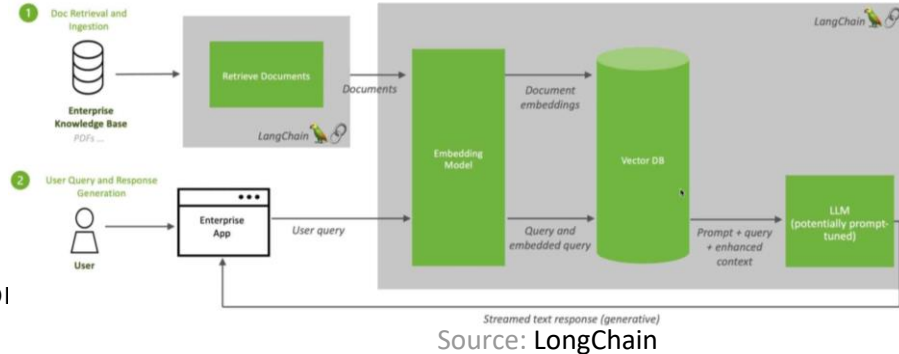
2. Omission of Key Information in the Prompt:

A lack of critical details can lead to incorrect responses.

Modern high-performance LLMs make these problems even more challenging to detect and correct.

Same story on Retrieval-Augmented Generation

- **Limitations of Information Selection:**
Selects and adds relevant information based on **semantic similarity**.
- **Drawbacks of Mechanical Information Addition:**
The LLM cannot effectively verify the accuracy of this information, which may lead to the generation of unverified content, resulting in errors.
- **Risks Beyond RAG Dataset Coverage:**
When the use case exceeds the scope of the RAG dataset, the added information would be unhelpful and misleading.



A vector space into which the words man, king, woman, and queen have been mapped. Source: [baeldung](#).

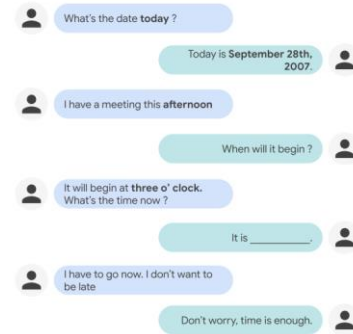
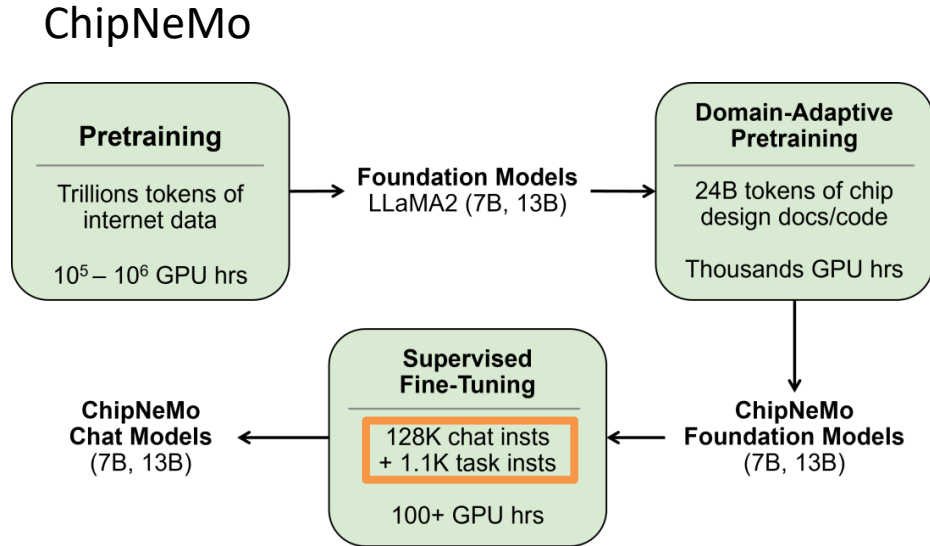


Key Trends in LLM Automation: Two Critical Points to Watch

1. **Simpler, Code-free Solutions:** Reduce coding requirements to make LLMs accessible for complex tasks.
2. **Enhanced Output Quality:** Implement strict quality control for accuracy, consistency, and relevance.

Leading us to train a domain-specific large language model

What Data Format Do We Need?



Such a format doesn't exist in open-source data.
Leading to **synthetic data** but **with real hardware designs**.

Behind Synthetic Data

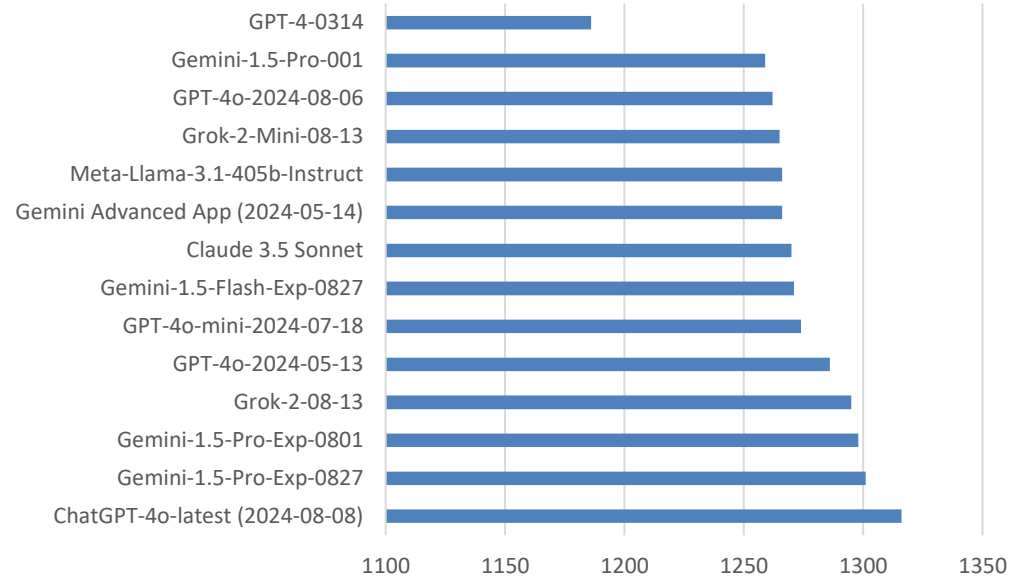
- General purpose LLMs are not hardware experts.
- But modern LLMs have impressive language ability

GPT-4's GRE grade

Simulated exams	GPT-4 estimated percentile	GPT-4 (no vision) estimated percentile	GPT-3.5 estimated percentile
Graduate Record Examination (GRE) Quantitative	163 / 170 ~80th	157 / 170 ~62nd	147 / 170 ~25th
Graduate Record Examination (GRE) Verbal	169 / 170 ~99th	165 / 170 ~96th	154 / 170 ~63rd
Graduate Record Examination (GRE) Writing	4 / 6 ~54th	4 / 6 ~54th	4 / 6 ~54th

Source:openai.com

LMSYS Chatbot Arena Leaderboard





Code Interpretation

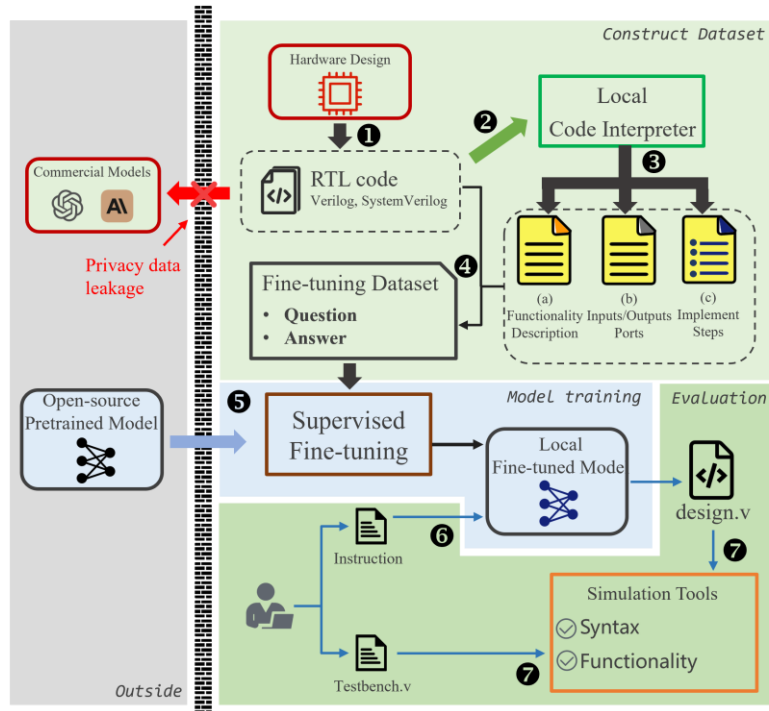
Previous work: Code Generation and Synthesis

- **Method:** Randomly combine prompts to generate hardware code.
- **Advantage:** Broad coverage.
- **Limitation:** Often lacks functional accuracy due to LLMs' software code bias.

Our Solution: Code Interpretation from Verified Sources

- **Method:** Interpret well-tested open-source hardware designs.
- **Advantage:** More reliable data.
- **Limitation:** Verified data is scarce and hard to collect.

Our Security Hardware Generation Solution



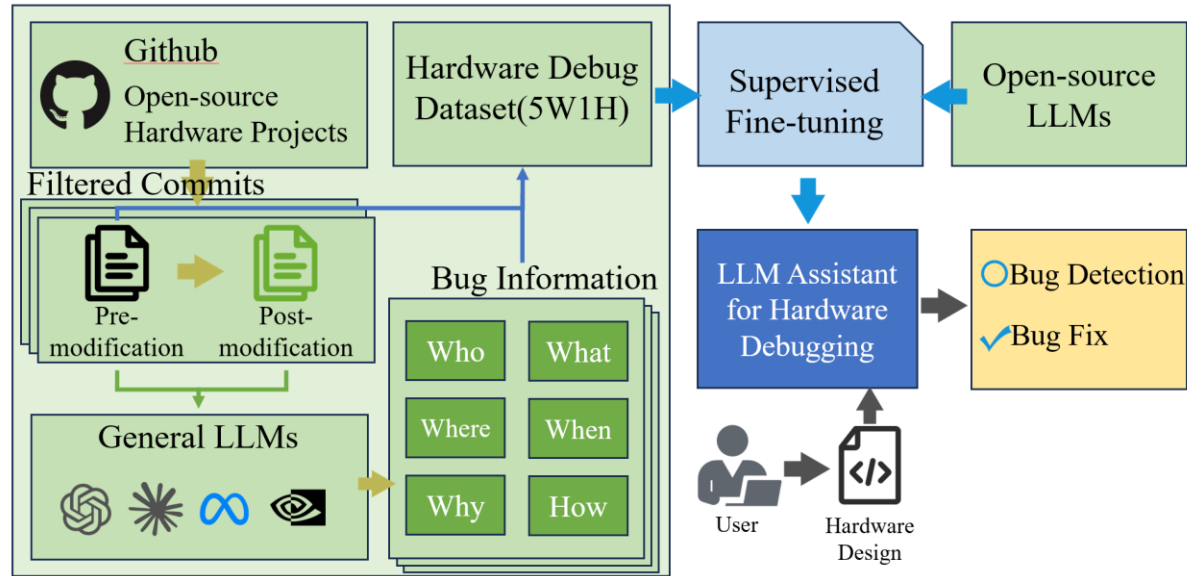
- Hardware Design: Verified Designs
- Local LLM as agent to explain the designs
- SFT for lowering the barrier

Submitted to AsianHOST 2024. *Intelligence In The Fence: Construct A Secure and Reliable Hardware Design Assistant LLM*. Shijie Li, Weimin Fu, Yifang Zhao, Xiaolong Guo, and Yier Jin

What do we achieve on this?

Model Type	Model	Params	VerilogEval (pass@k)						RTLLM v1.1(5-shot)	
			Eval-Machine(%)			Eval-Human(%)			Syntax-VCS(%)	Func(%)
			k=1	k=5	k=10	k=1	k=5	k=10		
Closed-Source	GPT4	N/A	60	70.6	73.5	43.5	55.8	58.9	100	65.5
	ChipNeMo	13B	43.4	N/A	N/A	22.4	N/A	N/A	N/A	N/A
	VerilogEval	16B	46.2	67.3	73.7	28.8	45.9	52.3	N/A	N/A
Open-Source	Verigen	16B	44	52.6	59.2	30.3	43.9	49.6	86.2	24.1
	Starcoder	15B	46.8	54.5	59.6	18.1	26.1	30.4	93.1	27.6
	RTLCoder	7B	62.5	72.2	76.6	36.7	45.5	49.2	96.6	48.3
Foundational Model	Llama-3-8B	8B	45.9	60.7	65.7	23.6	34.8	38.7	79.3	27.6
	Llama-3-70B-4bit	70B*4bit	65.5	71.7	74.2	41.2	46.7	48.7	86.2	51.7
Our solution	FT-Llama3-8B	8B	60.6	71.4	77.4	35.8	46.6	51.7	89.7	44.8
	FTLlama3-70B-4bit	70B*4bit	69.2	80.4	86.5	49.1	59.6	63.3	96.6	72.4

Utilizing Version Control for LLM Fine-Tuning



Submitted to *IEEE Transactions on Circuits and Systems I (TCAS I)*. A Generalize Hardware Debugging Approach for Large Language Models Semi-Synthetic Datasets. [Weimin Fu](#), [Shijie Li](#), [Yifang Zhao](#), [Kaichen Yang](#), [Xuan Zhang](#), [Yier Jin](#), and [Xiaolong Guo](#).

What do we achieve?

pass@1 OF BUG DETECTION TASK ON VALIDATION DATASET

Module #	Original Model								Fine-tuned model														
	GPT 4	GPT 3.5	Mixtral 8 × 7B	Codellama 70B	Llama2 70B	Code-llama 13B	Llama2 13B chat	Llama2 13B	GPT 3.5	Mixtral 8 × 7B	Codellama 70B	Llama2 70B	Code-llama 13B	Llama2 13B chat	Llama2 13B chat	Codellama 7B	Llama2 7B	Dolly 7B	Dolly 3B	Falcon 7B	StableCode 3B	Phi 1.5	Phi 2
1	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0.1	0	0	0	0.1	0	0
2	0.1	0	0	0	0	0	0	0	0.2	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0
3	0	0	0	0.1	0	0	0	0	0.3	0.2	0	1	0.1	0.2	0.4	0.2	0.1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0.7	1	0.1	0.2	0.2	0.1	0	0	0	0	0	0	0
5	0.1	0	0.2	0	0.2	0	0	0	0.3	0.5	0.2	1	0.1	0.2	0.2	0.2	0.2	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0.7	0.4	0	0	0.2	0.1	0	0	0	0	0	0
7	0.2	0.2	0	0	0	0	0	0	0.2	1	0.6	0.3	0	0	0	0.1	0	0	0.1	0	0	0	0
8	0.1	0.1	0.3	0.1	0	0	0	0	0	0	0	1	0.1	0	0	0	0.1	0	0	0.1	0	0	0
9	0	0	0	0	0	0	0	0	0.9	0.2	0	0	0	0.1	0.5	0.2	0	0.1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0.6	1	1	0.1	1	0.4	0.1	0.2	0.3	0	0	0	0	0.1	0
11	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0
12	0	0	0	0.1	0	0.1	0	0	0	0.2	0.2	0	0	0	0	0	0.2	0	0	0.1	0.1	0	0
13	1	0.2	0.4	0.5	0.2	0.1	0.1	0.1	0.2	1	0.5	0.9	0.3	0	0	0.1	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0.7	0.7	0.6	0.7	0.2	0	0	0	0	0	0	0	0
Overall	0.1	0.033	0.06	0.053	0.04	0.013	0.007	0.007	0.133	0.32	0.273	0.387	0.18	0.12	0.107	0.087	0.08	0.007	0.007	0.02	0.013	0.007	0



Beyond This

- Datasets train LLMs to **mimic** existing information.
- This learning process establishes **correlations** rather than causal relationships.
- As a result, even though our constructed datasets are secure, the LLMs trained on them may **not necessarily be secure**.



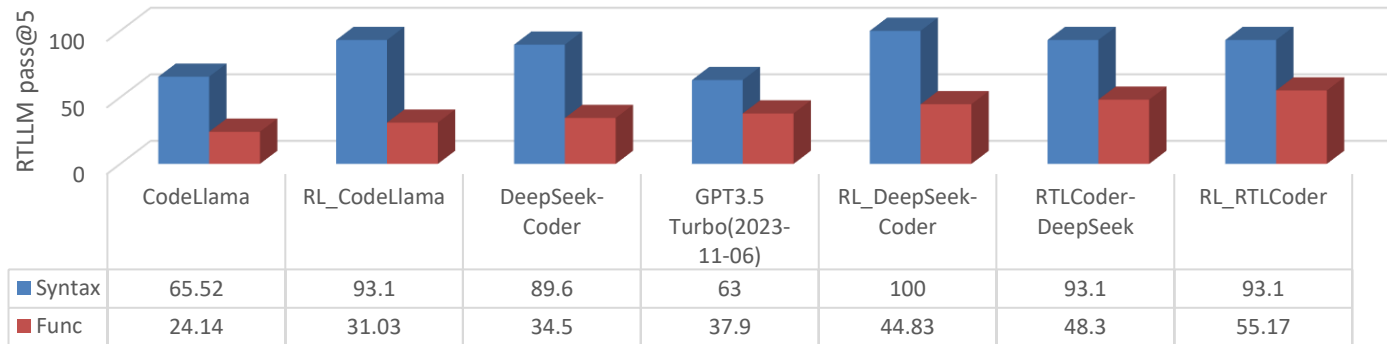
Introducing Reinforcement Learning

- Enables LLMs to improve decision-making through trial and error with feedback.
- By targeting specific security or performance goals, RF can enhance the quality of LLM-generated designs.

LLM could go beyond the **linear content** and **textual knowledge**.

A Case Study: Optimizing Text Similarity

- Introduced Abstract Syntax Tree and Data Flow Graph similarity to enhance the LLM's ability to generate syntax-compliant code. The results are as follows:



Accepted as a poster at CCS 2024. Enhance Hardware Domain Specific Large Language Model with Reinforcement Learning for Resilience.
Weimin Fu, Yifang Zhao, Yier Jin, Xiaolong Guo.

Case Study II: Introducing PPA Metrics

- PPA (Power, Performance, Area) metrics were introduced as evaluation parameters, enabling the LLM to consider attributes beyond text similarity in its generation process.

Design	Deepseek-Coder-6.7B									RTLCoder-DeepSeek								
	Perf			Power			Area			Perf			Power			Area		
	Original	RL Ver.	Improve	Original	RL Ver.	Improve	Original	RL Ver.	Improve	Original	RL Ver.	Improve	Original	RL Ver.	Improve	Original	RL Ver.	Improve
calendar	85	85	0.00%	4.128	4.128	0.00%	4.01	4.01	0.00%	85	85	0.00%	4.128	4.128	0.00%	4.01	4.01	0.00%
counter12	22	22	0.00%	3.61	3.61	0.00%	1.4	1.4	0.00%	22	22	0.00%	3.61	3.61	0.00%	1.4	1.4	0.00%
edge_detect	13	13	0.00%	0.686	0.684	0.29%	1	0.99	1.00%	13	13	0.00%	0.686	0.684	0.29%	1	0.99	1.00%
pe	193	193	0.00%	36.884	36.884	0.00%	19.82	19.82	0.00%	193	193	0.00%	36.884	36.884	0.00%	19.82	19.82	0.00%
right_shifter	37	37	0.00%	2.182	2.182	0.00%	2	2	0.00%	37	37	0.00%	2.182	2.182	0.00%	2	2	0.00%
synchronizer	48	48	0.00%	1.244	1.243	0.08%	2.6	2.6	0.00%	48	48	0.00%	1.244	1.244	0.00%	2.6	2.6	0.00%
RAM	309	231	25.24%	2.063	2.063	0.00%	4.42	4.38	0.90%	1581	212	86.39%	3.762	3.762	0.00%	5.76	5.75	0.17%
width_8to16	126	94	25.40%	1.558	1.866	-19.77%	5.61	5.6	0.18%	130	126	3.08%	1.161	2.24	-92.94%	5.61	5.6	0.18%
adder_8bit	9	9	0.00%	5.753	5.751	0.03%	5.22	5.2	0.38%	9	9	0.00%	5.753	5.752	0.02%	5.22	5.2	0.38%
adder_16bit	17	17	0.00%	11.269	11.139	1.15%	10.01	10	0.10%	17	16	5.88%	12.324	10.174	17.45%	10.01	10.01	0.00%
radix2_div	N/A	122	100.00%	N/A	0.652	N/A	N/A	7.42	N/A	167	94	43.71%	0.187	0.433	-131.55%	3.4	4.1	-20.59%
traffic_light	61	33	45.90%	3.003	4.535	-51.02%	2.81	2.6	7.47%	33	33	0.00%	4.2	4.196	0.10%	2.81	2.6	7.47%
freq_div	21	21	0.00%	2.358	2.358	0.00%	1	1	0.00%	38	38	0.00%	2.82	2.82	0.00%	1	1	0.00%
accu	198	129	34.85%	2.579	3.745	-45.21%	4.42	4.42	0.00%	121	159	-31.40%	1.264	3.691	-192.01%	4.41	4.42	-0.23%
fsm	15.5	15	3.23%	0.541	0.247	54.34%	0.8	0.8	0.00%	15	16	-6.67%	0.308	0.265	13.96%	0.8	0.8	0.00%
JC_Counter	443	443	0.00%	129.119	129.119	0.00%	13.22	13.22	0.00%	443	443	0.00%	129.119	129.119	0.00%	13.22	13.22	0.00%
multi16bit	336	336	0.00%	9.76	4.605	52.82%	10.16	8.63	15.06%	347	337	2.88%	5.669	1.002	82.32%	13.66	13.66	0.00%
multibooth8bit	301	257	14.62%	0.563	0.577	-2.49%	7.05	7.04	0.14%	400	400	0.00%	2.974	2.974	0.00%	7.05	7.05	0.00%
multiplier8bit	157	113	28.03%	3.932	8.321	-111.62%	3.62	3.62	0.00%	89	67	24.72%	5.046	1.691	66.49%	3.61	2.41	33.24%
multiplier8bit	458	167	63.54%	12.729	16.533	-29.88%	7.25	7.24	0.14%	577	N/A	#VALUE!	12.133	N/A	N/A	7.25	N/A	N/A
parallel2serial	43	43	0.00%	4.293	4.293	0.00%	2.2	2.2	0.00%	31	28	9.68%	2.255	2.249	0.27%	1.6	1.6	0.00%
pulse_detect	14	12	14.29%	0.713	0.416	41.65%	0.8	0.8	0.00%	14	15	-7.14%	0.258	0.296	-14.73%	0.8	0.8	0.00%
serial2parallel	88	88	0.00%	2.264	2.499	-10.38%	2.6	2.6	0.00%	88	64	27.27%	2.499	0.569	77.23%	2.6	2.6	0.00%
signal_generator	29	24	17.24%	3.942	4.065	-3.12%	1.41	1.4	0.71%	24	24	0.00%	4.069	4.069	0.00%	1.4	1.4	0.00%
alu	370	356	3.78%	16.397	12.173	25.76%	21.68	21.6	0.37%	370	363	1.89%	12.529	12.441	0.70%	21.61	21.56	0.23%
div_16bit	134	16	88.06%	14.293	4.167	70.85%	11.36	9.6	15.49%	N/A	32	N/A	N/A	33.139	N/A	N/A	11.35	N/A
adder32bit	N/A	16	N/A	N/A	10.689	N/A	N/A	13.01	N/A	32	11	65.63%	18.425	4.788	74.01%	18.62	18.61	0.05%
adderpipe	731	583	20.25%	53.239	52.206	1.94%	39.47	39.43	0.10%	729	781	-7.13%	56.17	54.517	2.94%	39.47	39.47	0.00%
asynfifo	N/A	485	N/A	N/A	9.223	N/A	N/A	16.7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Accepted as a poster at CCS 2024. Enhance Hardware Domain Specific Large Language Model with Reinforcement Learning for Resilience.
 Weimin Fu, Yifang Zhao, Yier Jin, Xiaolong Guo.



Our willing

- More situation
- More reliable

Thank you for listening

- If you are interested, don't hesitate to get in touch with me at weiminf@ksu.edu or my advisor, Prof. Xiaolong Guo, at guoxiaolong@ksu.edu.

Acknowledgments

- I thank Prof. Guo for allowing me to explore my research direction and for his unwavering support.
- These works are supported by the National Science Foundation (2340949) and ARISE(2148878)

