Statement of Research Interests, Accomplishments, and Vision — Jason Blocklove

Research Vision

Large language models (LLMs) are continuously lowering the bar for the design of custom hardware, and, through those same methods, for attacking it as well. My primary research studies both sides of this paradigm: leveraging LLMs for more accessible custom hardware design by non-experts, and evaluating the security of those hardware designs in the new generative AI landscape. By determining best-use practices and developing frameworks for LLMs to enable more widely-accessible custom hardware designs, I seek to enable a broader scope of fields to benefit from designing their own hardware as needed. Prior to the advent of LLMs, purpose-designed hardware was difficult to produce due to the niche skillset necessary to develop it. With the potential to generate significantly more hardware, so too comes the realities of generating more *insecure* hardware and enabling bad actors to more easily generate *malicious* hardware. As such, I seek to both understand the ramifications of these potential growing security threats, as well as examine how LLMs can help mitigate security vulnerabilities in hardware as well.

Recent and Ongoing Research

LLM-Aided Digital Hardware Design: ChatGPT was released shortly after I began my Ph.D. and provided an exciting new avenue for both general use and research. However, there were no defined best practices for its use and initial experiences for hardware design were quite poor. Some limited successes, though, led to the formulation of my initial research questions: "How can experienced hardware designers best make use of LLMs?" and "How can LLMs make hardware design more accessible for those outside of the field?"

In "Chip-Chat," my initial work which taped out the world's first LLM-designed hardware [MLCAD '23] (cited over 400 times), my goal was to examine how effectively a skilled hardware engineer could prompt an LLM to create a novel design with real-world constraints. This targeted GPT-4, the most advanced model at the time, and TinyTapeout, a project for inexpensively taping out hardware with significant design constraints on size, speed, and IO. Guided by an experienced designer only using natural language prompts, GPT-4 was made to generate a novel accumulator-based instruction set architecture (ISA) including initial specifications, all Verilog for implementation, and a Python assembler for use in testing. This testing was still done by the engineer. This novel design was successfully taped out and validated in hardware, demonstrating that LLMs, despite difficulties with hardware design, could act as force-multiplying tools by experienced engineers.

While these results were exciting, they didn't yet indicate if LLMs could make hardware design more accessible to those who lack hardware design experience. To tackle the question of how LLMs could enable non-experts to design hardware, I developed a structured LLM framework using Verilog compiler feedback and three levels of engineer-driven feedback to determine how well LLMs could generate both functional designs and their tests and how much human expertise was needed [LAD '24]. Once a design and its testbench were generated, they were compiled and tested and any errors were fed back to the LLMs under test, with human feedback being added once the LLM failed to fix the issues multiple times. The human feedback was given in stages, getting more explicit as the errors persisted. Through this I was able to track how different LLMs fared with different benchmarks as well as how much hardware design expertise was necessary to generate functional benchmarks. Results were promising at the time, though LLM-generated testbenches were often lacking reasonable test coverage and failed to properly test designs even if they were correct.

The structured feedback method still required a comprehensive understanding of the design and tools to provide explanations of the errors when necessary. In my following work I devised *AutoChip* [Preprint '24], [TODAES '25] (cited 275 times across preprint and publication), the first fully-automated agentic hardware design framework using LLMs, which aimed at removing the need

for prior domain knowledge. Based on the same structure as the previous work, AutoChip relies on both a design prompt and a verbose testbench with quality feedback to form iterative loop where Verilog is generated and then corrected. I found that with just a simple iterative setup the LLMs often got stuck going down incorrect or unhelpful paths when correcting errors. To address this I expanded AutoChip to use a tree-search method, generating multiple candidate responses at each step and using the highest quality response for feedback. This led to an 84% success rate on the VerilogEval-Human benchmark set, a standard set of benchmarks for LLM-based hardware design. Full automation from natural language begins to crack open the door for accessible hardware designs by those without the niche expertise. AutoChip, and the artifacts from all of my other works, is *fully open-source* in line with my goal of making hardware design accessible.

During these studies, I found that efficiency was often overlooked. Newer models have been rapidly increasing in complexity, pushing up energy use and cost, and challenging the goal of accessibility. In AutoChip I found that the most advanced model, GPT-4o, made better use of the tool feedback and gave the best results of the models tested, but was significantly more expensive. This led to a new research question: *"How can we increase quality while decreasing the use of the most complex models?"* I examined this by mixing models within AutoChip, generating most of the candidates with smaller models, and querying GPT-4o only if the maximum search depth had been reached. Simple problems now only used the energy needed, and problems requiring larger models only used them once a level of function has already been achieved. These experiments showed an 82.8% reduction in cost for 89.2% of the success rate of GPT-4o alone. I aim to continue this efficiency analysis, as it has a significant impact on this technology being widely usable.

I, along with my collaborators, have a book accepted for publication by Springer on this topic titled *"From Quips to Chips: Toward Chip Design with Generative AI"*.

LLMs and Hardware Security Implications: With LLMs making hardware design more accessible it raises two notable concerns with security: LLMs making it easier to generate malicious hardware, and ensuring that the new hardware being developed with LLMs is itself secure.

The first step to protecting a system is to understand what the potential threats are. I designed and organized a student competition called the AI Hardware Attack (AHA!) Challenge which tasks teams with using generative AI to design and implement vulnerabilities and hardware Trojans into open-source designs as a means of seeing how LLMs could make malicious hardware design available to bad actors of various skill levels. The competition structure was used to leverage the creativity of student teams will different backgrounds to gather a diverse collection of results. The challenge is currently running its sixth iteration in four years across NYU's CSAW, HOST, and DAC. Results from the first two competitions showed that LLMs were able to significantly "lower the bar" for bad actors, allowing teams with no prior hardware design or security experience to create effective attacks which evaded detection by the core projects' existing verification methods [IEEE S&P Mag '25]. Following the initial competitions, the challenges have evolved alongside LLMs, requiring more complex vulnerabilities, increased LLM automation and agentic usage, and most recently FPGA bitstream reverse engineering aimed at specific hardware in a simulated third-party IP attack.

With LLMs shown to simplify attacks on hardware designs, security analysis becomes a key focus. Recently there has been an effort to standardize the idea of and process of identifying "security assets" in hardware. Industry-scale designs are often massive projects, so these assets are key to providing a map for security analysis; however, security-centric designs can consist of hundreds of assets requiring tracing and analysis. I have sought to determine how LLMs can simplify this process by providing a triage of potential security bugs. It has become clear with the proliferation of LLMs that they are not perfect and should not necessarily be a generic replacement for already well-formed task-focused tools, especially in security contexts. As such, I developed a

novel system which leverages existing static-analysis tools together will LLMs to contextualize and triage security bugs in large hardware designs. The static analysis tools are used to identify assets based on IEEE guidelines and identify common weakness enumerations (CWEs) related to those assets. This information provides context that gets passed to an LLM agent, along with additional context documentation for the hardware design itself, which creates security annotations for each asset-CWE pair. These annotations are passed to a second LLM agent which triages each asset based on the potential overall threat and provides a complete security engineering report. This research is ongoing with support from Intel and includes analyses of per-task LLM selection, aimed at optimizing efficiency and effectiveness of different models at each tasks. I have been leading a team of three other Ph.D. students in this work, one from the University of Calgary and two from the University of New South Wales, Sydney.

Future Work

LLMs have continued to improve at an extremely rapid pace, creating new questions and angles faster than we've been able to conduct research on them. My goals for future work begin with expanding upon the study of efficient use of LLMs, as started in AutoChip. LLMs have continued to grow in size and complexity and as such have become increasingly costly and power hungry. Through the context of hardware design I aim to identify best practices for both model selection given a specific task, as well as ways of using the most capable models more efficiently. I aim to further develop this work by exploring more use of local models and small language models (SLMs), which not only stand to improve model use efficiency, but also can be positioned to help work in industries where remote LLMs create concerns of proprietary data leakage into these models.

Hardware verification also remains an issue, especially when considering the potential use of LLMs by non-experts to create hardware designs. While research has been done on improving testbench and formal assertion generation by LLMs, there is still a gap for experts in their own fields verifying hardware using systems to them. I seek to work with experts in other fields who might benefit from custom hardware to not only better determine how LLMs can be used to democratize the hardware design process but also the hardware verification process. Within ECE I have recently given a tutorial on LLM-based hardware design at BCICTS, a primarily RF-focused venue, with the goal of identifying specific problems that could benefit from custom hardware but currently lack the skills. My goal in this line of research is to make hardware designing as accessible as possible, so those in other fields may take advantage of the benefits of custom hardware.

In hardware security, while we have identified that LLMs are able to generate functional hardware designs, little research has been done to this point on examining whether they create *secure* hardware designs as well. With the increased capabilities of complex agentic workflows being able to develop more complex multi-module designs, the potential for insecure designs to be generated grows as well. While some research was conducted on secure code generation in the early days of LLMs, the focus on hardware was minimal and LLMs have improved drastically since. Along with continued efforts in security analysis of complex designs, I aim to examine the extents to which modern LLMs may generate hardware with unintentional security vulnerabilities (as opposed to the intentional vulnerability additions from the AHA! challenges), as well as what prompting strategies impact these generated vulnerabilities from the start.

Conclusion

My research vision has up to this point focused primarily on examining how generative AI can be used to simplify the creation of hardware, with the overarching goal of making hardware more accessible to any field which could use it, through both improving the results and the efficiency of LLM use. This focus is has shifted more towards expanding the role of LLMs in hardware security as I've seen more how the same processes that make hardware design easier also make insecure or malicious hardware easier to create.

References

[MLCAD '23] J. Blocklove, S. Garg, R. Karri, and H. Pearce, "Chip-Chat: Challenges and Opportunities in Conversational Hardware Design," in *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, Sep. 2023, pp. 1–6. DOI: 10.1109/MLCAD58807.2023.10299874

[LAD '24] J. Blocklove, S. Garg, R. Karri, and H. Pearce, "Evaluating LLMs for Hardware Design and Test," in *2024 IEEE LLM Aided Design Workshop (LAD)*, Jun. 2024, pp. 1–6. DOI: 10.1109/LAD62341.2024.10691811

[Preprint '24] J. Blocklove, S. Thakur, H. Pearce, B. Tan, S. Garg, and R. Karri. "AutoChip: Automating HDL Generation Using LLM Feedback." arXiv: 2311.04887 [cs], pre-published.

[TODAES '25] J. Blocklove, S. Thakur, B. Tan, H. Pearce, S. Garg, and R. Karri, "Automatically Improving LLM-based Verilog Generation using EDA Tool Feedback," *ACM Trans. Des. Autom. Electron. Syst.*, Mar. 26, 2025, ISSN: 1084-4309. DOI: 10.1145/3723876

[IEEE S&P Mag '25] J. Blocklove, H. Pearce, and R. Karri, "Lowering the Bar: How Large Language Models Can be Used as a Copilot by Hardware Hackers," *IEEE Security & Privacy*, pp. 2–12, 2025, ISSN: 1558-4046. DOI: 10.1109/MSEC.2025.3600140