Pengju LIU | **E-mail**: pliuas@connect.ust.hk | **Tel**: (+852) 5487-3182

EDUCATION

M.Sc. in The Hong Kong University of Science and Technology (HKUST) Hong Kong, China
--- Program: Integrated Circuit Design Engineering Sept, 2023 – Jun, 2024
--- GPA: 3.841/4.30 (first semester's GPA 4.225/4.30)

B. Eng in University of Electronics Science and Technology of China (UESTC) Chengdu, China
---Major: Microelectronic Sept, 2019 – July, 2023
---GPA: 3.77/4.00
---Average score: 86.03/100

PUBLICATIONS

[C1] Yiyao Yang, Teng Fu, **Pengju Liu**, Mengnan Qi, Chenyang Lv, Ji Li, Xuhong Zhang, Zhezhi He, "HAVEN: Hallucination-Mitigated LLM for Verilog Code Generation Aligned with HDL Engineers", IEEE/ACM Proceedings Design, Automation and Test in Europe (**DATE**), Lyon, France, Mar. 31-Apr.02.2025.

[C2] **Pengju Liu**, Yiyao Yang, Teng Fu, Ji Li, Xiangyu Yue, Qiang Xu, Xuhong Zhang, Zhezhi He, "MMVerilog: An Automated LLM Framework for Top Module Design with Submodule Relationship Capturing and Triggering-Aware Annotator", submitted to DAC2025.

PROJECT EXPERIENCE

COOL-to-MIPS Compiler Implementation and Execution on SPIM MIPS Simulator
Direct: Prof. Fredrik Kjolstad(Stanford, CS143) Mar. 2024 - present

- Utilized the Flex tool to design the lexer, which transformed input COOL code into well-defined lexical tokens. This initial step laid the foundation for subsequent parsing activities.
- Employed the Bison tool to design the parser, which organized the lexical tokens into a structured and hierarchical Abstract Syntax Tree (AST), facilitating further stages of type checking and code generation.
- Implemented the type checking for generated AST, which consists of two traverses of AST, one for checking the correctness of inheritance graph, the other for annotating the proper type of each expression of AST for code generation stage.
- Currently in progress is the implementation of code generation functionalities. The former ensures the type safety of COOL code, while the latter translates COOL code into equivalent MIPS assembly code.

Implementation of binding based on Left Edge Algorithm and Verilog generation for light-HLS tool
Director: Prof. Wei ZHANG(HKUST) Apr. 2024 – Jun. 2024

- Learned from bambu HLS, which is a famous template for constructing the HLS tool, and mastered the operations and optimization of binding stage, and how binding stage interacts with other parts of bambu, such as frontend, scheduling, technology and so on.
- Utilized LLVM architecture to implement register binding and operator binding, using Left edge algorithm and explored alternative algorithm, like Chordal Graph Coloring Algorithm to decrease register's number when facing structure like if-else basic block.
- Currently working on optimizing the Verilog code generation phase of Light-HLS, based on an initial version, to resolve conflicts in the FSM and reduce resource utilization waste.

Implementation of a Multi-Modal LLM for Capturing Topology from Graphs
Director: Prof. Zhezhi HE(SJTU) Oct. 2024 - Oct. 2024

- Developed a framework to enable a large language model (LLM) to effectively capture topology information. This involved creating a Graph Neural Network (GNN) using Torch Geometric to extract topological data. Proposed an alignment module to integrate graph and text modalities, ensuring seamless interaction between the GNN and the backbone LLM.
- Established a comprehensive training framework capable of processing the original dataset. This framework automatically handles text modality inputs, attention masks, labels, and graph modality inputs for efficient training of the multi-modal LLM.

Implementation of Verilog e-graph optimizer based on paper *"ROVER: RTL Optimization via Verified E-Graph Rewriting"*
Director: Prof. Zhezhi HE(SJTU) Jul. 2024 – Aug. 2024

- Developed a framework for Verilog optimization focused on PPA, based on the open-source Egg framework and the "ROVER" paper. This framework preprocesses the initial Verilog code, then parses a JSON file into VeriLang, a new IR used in "ROVER".
- For the mid-phase, the Egg framework is utilized to rewrite Verilog using established rules and to explore the entire design space. Ultimately, one implementation is selected from the vast design space, with continuous verification throughout the process.
- The framework is capable of optimizing PPA based on various cost functions.

SKILLS

Programming: C/C++, python, Verilog, Shell, VHDL, Rust

Tools: Pytorch, LLVM, Vitis HLS, Matlab, Vivado, Cadence virtuoso, Synopsys, LaTex

Language: Chinese (native language), English (Fluent, IELTS 7.0)