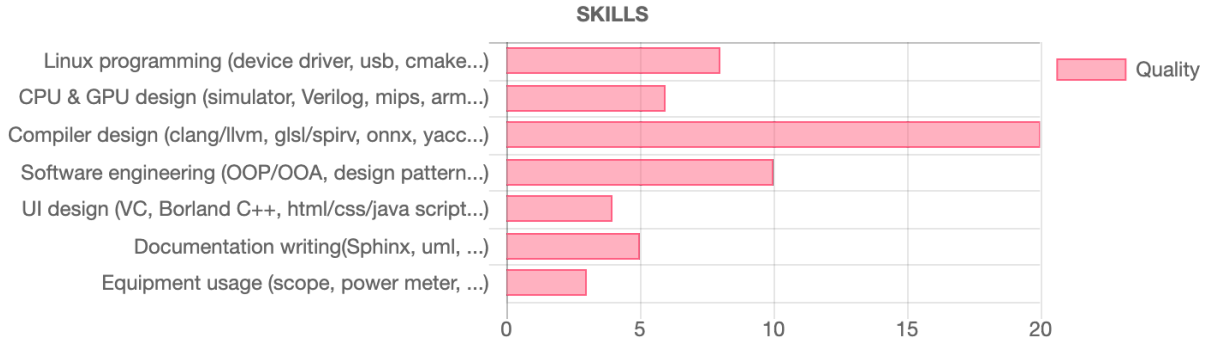
I am a compiler developer with solid experience in LLVM CPU and GPU backends, the LLD linker, NPU/ONNX, C++, OpenGL/GLSL, simulators, and more. I enjoy working on compilers and related technologies.

RESUME

QUALIFICATION

Over 20 years of experience in C/C++ programming, with 13 years focused on compiler



MY OPEN SOURCE PROJECT

I'm proud that my work is featured in the official LLVM documentation under http://llvm.org/docs/tutorial/#external-tutorials.

Tutorial: Create an LLVM Backend Compiler http://jonathan2251.github.io/lbd/index.html

Tutorial: Create an LLVM Backend Toolchain http://jonathan2251.github.io/lbt/index.html

The Concept of a GPU Compiler http://jonathan2251.github.io/lbd/gpu.html

EDUCATION

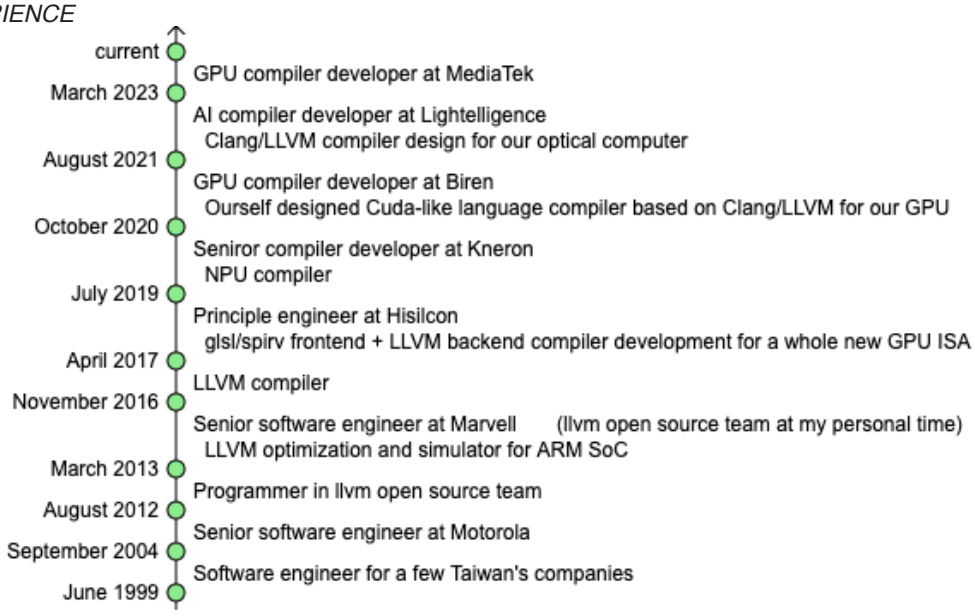
Master's Degree, Information Science, National Taiwan Normal University (國立台灣師範大學), Taipei — June 1999

Bachelor's Degree, Industrial Engineering, National Taiwan University of Science and Technology (國立台灣科技大學), Taipei — June 1994

LICENSE

National Senior Technician Certificate in Information Technology (國家高考資訊技師), Taiwan — 1995

EXPERIENCE



Master's Thesis

The Researches of Column Sort and Related Problems

Conference Paper: Search for "行排列法簡化步驟之研究" on the above link.

PhD Study Proposal

The Researches of Sorting Network and Related Algorithm

OTHER WORK

Took a course in image processing and developed Jpeg decoder

Web and javascript: As my resume and my personal web site

Graphivz: as some graph diagrams used in this CV. Source code: mywork_1.gv and study_and_apply.gv

ACHIEVEMENT

Lightelligence

Developed backend compiler for Lightelligence's optical NPU based on RISC-V includes:

1. Built a complete RISC-V compiler toolchain using GCC, LLVM, and QEMU/Gem5 from open-source projects. Evaluated RISC-V vendors and pricing negotiations, leveraging our in-house ability to build the RISC-V toolchain from open source.

2. Led software development for the Aurora hardware product and personally programmed the compiler backend.

3. Developed the TaskGraph component in the C++ compiler and its interfaces to the Runtime module, enabling Deep Learning Graph features on our platform.

Biren

Developed GPU code generation for tensor instructions and handling of usharpid.

Optimized GPU performance and fixed related bugs.

Proposed solutions for parallel processing in our CUDA-like language async{...}.

Kneron

Re-implemented the top two layers of our NPU compiler to support a common graph data structure on different types of NPU.

Developed compiler input interfaces to support encrypted ONNX models and configuration file formats.

Validated solutions for MLIR support integration.

Hisilcon

Scope of GPU Compiler Work:

Compared our GPU compiler code with the ARM-licensed version; approximately 20% of the frontend and 50% of the backend were modified, based on lines of code.

My contributions:

Independently implemented the compiler frontend and LLVM backend for approximately 80% of texture-related APIs, based on the OpenGL ES 3.2 specification, including documentation.

Provided guidance and support to other engineers on the remaining 20% of texture-related APIs, reviewed their code, and collaborated with the texture architecture lead.

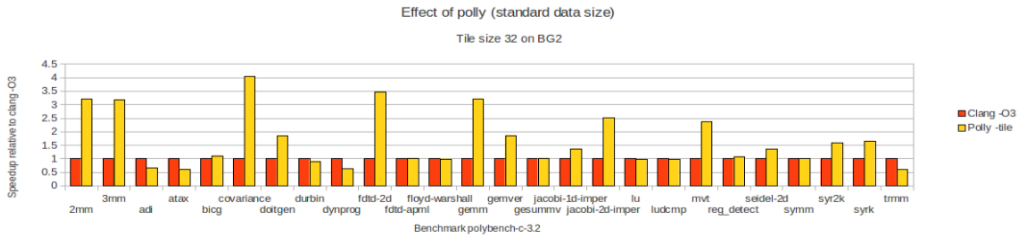
Implemented a Prefetch-Sample optimization, allowing 2D texture sampling instructions to be executed by the driver before GLSL shaders are loaded or run.

Developed compiler support for our GPU's load/store operations in Vulkan, including RGBA fixed-point formats (32, 16, 11, 10, and 2 bits), with handling for NaN and Infinity values. Also authored related documentation.

Marvell

Developed a semi-automated software system to run benchmarks and generate reports for the GCC toolchain.

Demonstrated the use of Polly, a loop optimization framework, and introduced the concept of the polyhedral optimization model for improving both LLVM and GCC toolchains at Marvell.



Implemented a co-simulator for several of Marvell's ARM-based 64-bit CPUs.

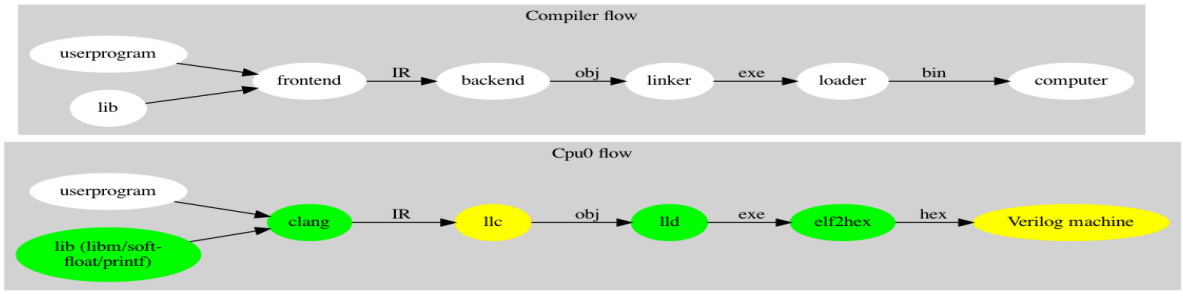
Proposed and implemented a domain-specific language (DSL) within the co-simulator, significantly reducing the amount of C++ code required for system verification.

Replaced the existing Make-based build system with CMake for the CSim project.

Benefit: CMake offers a simpler and more cross-platform solution compared to Make.

LLVM Open Source Project

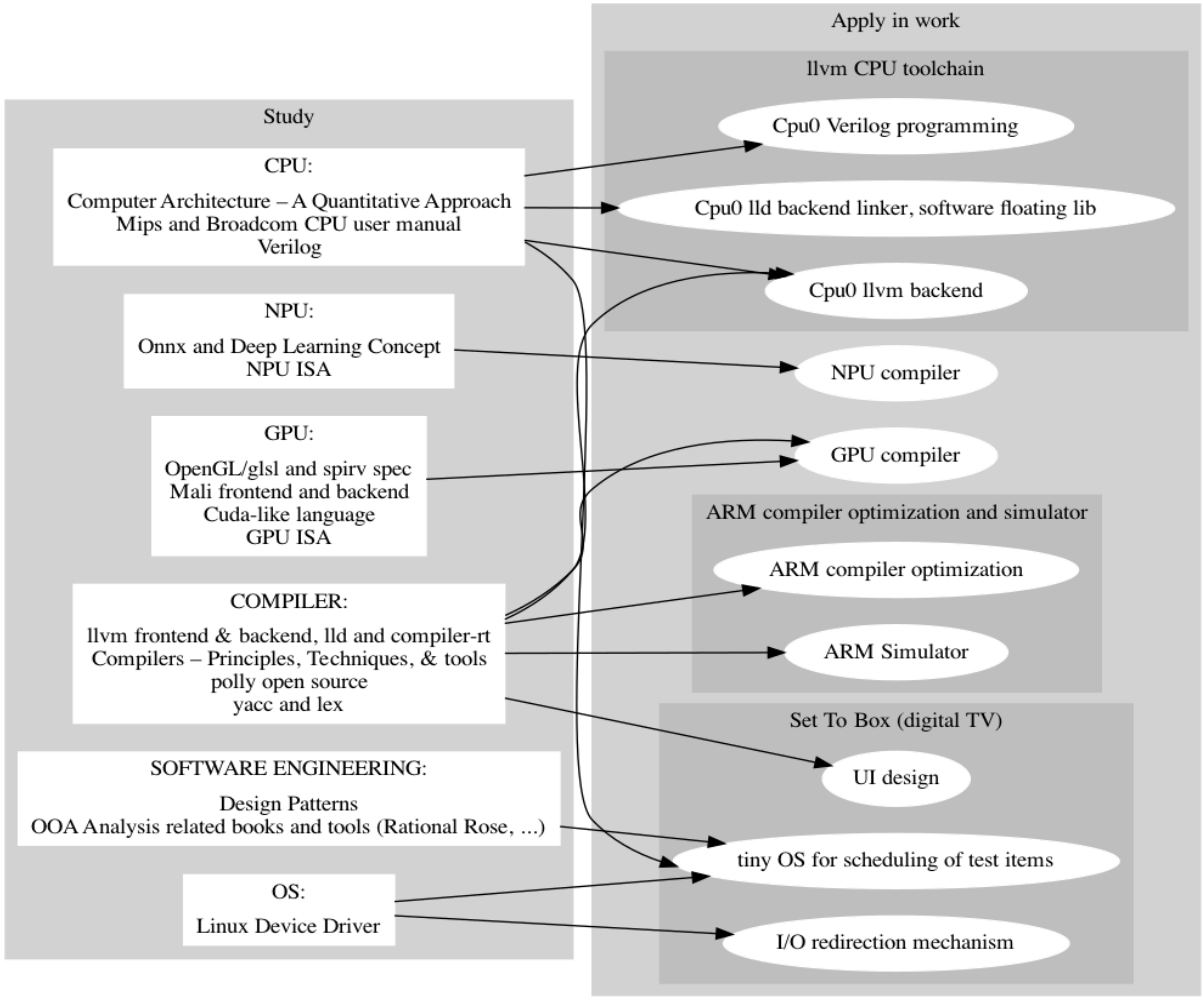
The lower half of diagram below illustrates the workflow of my LLVM backend. The yellow and green sections represent components I implemented, as documented in my tutorials.



Motorola

Developed the software framework for Set-Top Box systems.

Learning Beyond School and Applying It at Work



References

[Recommendation Letter from Former Manager: https://jonathan2251.github.io/ws/en/RL_Marvell.pdf]