Andy Wanna

404-660-0194 — awanna3@gatech.edu — Web Page — LinkedIn — GitHub

Summary — Computer engineering PhD Student focusing on improving hardware design automation and optimization for modern AI and cryptographic workloads. I aim to develop and apply the skills gained through research to practical industry tools and solutions in hardware processor design.

Skills

Programming C/C++, Python, Rust, CUDA
Hardware SystemVerilog, HLS, Vitis, Quartus
Machine Learning PyTorch, Nvidia TensorRT, Quantization
Software Git, AWS, SQL

Education

Georgia Institute of Technology **Aug 2024 – May 2029**
Ph.D. in Electronic and Computer Engineering

Imperial College London **Oct 2020 – Jul 2024**
Masters of Engineering in Electronic and Information Engineering - 1^{st} Class Honours (4.0 GPA)
Dean's List 3^{rd} Year (Top 10%), Dean's List 4^{th} Year (Top 5%)

Technical Experience

Numerical Hardware Engineer Intern at Intel **Jun 2024 – Aug 2024**
- Automated mixed-precision floating point multiplier design optimization in Rust using E-Graphs
- Investigated glitch power minimization in hardware designs using Integer Linear Programming
- Explored techniques and methods for optimal dot-product hardware units

FPGA Engineering Co-Op at Quantum Motion **Apr 2023 – Oct 2023**
- Designed and integrated bespoke signal generator for High-Speed Qubit Feedback on an FPGA
- Built custom Python code-base using PYNQ and QICK interface with the FPGA
- Programmed RP2040 MicroController to communicate using TCP/IP over ethernet connections in C

Analogue and Digital IC Validation Intern at Quantum Motion **Jul 2022 – Oct 2022**
- Validated operation of Ring Oscillator hardware on silicon chips
- Analysed transistor properties across a range of chips, at room and cryogenic Temperatures (2K) with Python

Research Experience

Hardware Accelerator Generation and Optimization for Cryptographic Primitives **Aug 2024 – Present**
Georgia Institute of Technology
- Automatically applying HLS Optimizations to Cryptographic Primitives, using E-Graphs to generate formally verified cryptographic accelerators for resource-constrained hardware
- Developing surrounding C-to-RTL HLS workflow to guarantee the correctness and optimal accelerator PPA

Quantifying and Automating Interpretability for Deep Learning **Oct 2023 – Jun 2024**
Imperial College London - Final Year Project - Primary Author - Submitted to AAAI 2025
- Designed metric to quantitatively evaluate the interpretability of a deep learning model, enabling trade-off analysis
- Developed an automated workflow to add interpretability to CNNs, achieving up to an 84% inference latency improvement over the manual INN model
- Demonstrated comparable interpretability results on standard CNNs with state-of-the-art GradCAM methods

OptiMult - Multiplier Optimization via E-Graph Rewriting **Dec 2022 – Sep 2023**
Imperial College London - Primary Author - Presented at ASILOMAR 2023 - Available Here
- Created tool in Rust to compile hardware arithmetic expressions into optimized representations for area and latency
- Demonstrated up to a 46% latency reduction in squarer circuits and 9% latency reduction in general multiplication against industry standard logic synthesis tools

Notable Projects

Deep Learning Activation Function Hardware Accelerators in SystemVerilog **Jan 2024 – Mar 2024**
- Designed and Verified 4 – 16 bit parameterized implementations of Softmax and element-wise activation functions
- Integrated activation functions into a re-configurable, dynamically scheduled hardware architecture

Deep Learning Models in PyTorch - U-Net, VAE Transformer, DCGAN **Jan 2023 – Mar 2024**
- Implemented a U-Net CNN architecture to perform image classification and segmentation on brain scans
- Analyzed a VAE transformer architecture on generating and classifying the MNIST dataset
- Investigated Class Conditional & Wasserstein-GP DCGAN architectures on generating images in the CIFAR-10 dataset

Cosine Accelerator on FPGA in SystemVerilog **Jan 2023 – Mar 2023**
- Reduced the latency of a vector function by 77% on an FPGA using an optimized CORDIC cosine block in SystemVerilog
- Created customized hardware floating-point arithmetic blocks, with low-level programming in C for further speed-ups