Tai Williams

twilliamsa776@gmail.com | linkedin.com/in/tai-a-williams | github.com/TWilliamsA7

Education

University of Central Florida, Burnett Honors College — Orlando, FL
Bachelor of Science in Computer Engineering — *August 2024 – May 2028*

Experience

ASIC Physical Design Intern — May 2026 – August 2026
NVIDIA — *Santa Clara, CA*

- Reduced pre-PnR layout iteration time from days to hours on large, hierarchical chiplet designs, by architecting a Python-based LLM agent that autonomously diagnosed tool and script errors and resolved them using its own toolset, without hardcoded logic.
- Prevented downstream integration failures across **5** blocks (4 LPU, 1 CPU), by building a Python dashboard that automatically flagged DRC, PG, and RTL-level design violations on every incoming RTL revision and ASIC-PD fix.
- Enabled placement convergence on early-stage designs where the vendor full-placement flow frequently failed to run or was difficult to configure, by developing a tuned heuristic optimizer that used extracted design data to generate a cheap, accurate placement estimate for use alongside physical synthesis iterations.
- Unified fragmented, agent-unsafe constraint files across **3** IPs, previously spread across incompatible formats, into a single Python-based internal framework supporting natural-language-driven automated editing, cutting engineer setup time by **80%**.
- Cut pipeline runtime **7x** and memory footprint **90%** through parallelization, caching, and cheap-but-accurate heuristic approximations, unlocking additional parallel execution critical for scaling to larger designs.

Undergraduate Learning Assistant — August 2025 – Present
University of Central Florida — *Orlando, FL*

- Grew average weekly session attendance **4.5x** by leading **5+** weekly live **C** programming aid sessions and open office hours for **200+** students, diagnosing common pitfalls and fostering peer collaboration.
- Developed comprehensive **C** programming study guides with targeted challenges on data structures, pointers, and recursion, correlating with stronger exam performance on covered topics and streamlining student preparation.

Projects

RISC-V CPU Emulator | *C++, CMake, Python* — February 2026 - May 2026

- Accomplished a high-fidelity RISC-V instruction set simulator as measured by a **100%** pass rate across official RISC-V rv32 architectural test suites, by implementing a modular instruction decoder and execution engine in **C++**.
- Architected an Sv32 Virtual Memory Management Unit (MMU) as measured by successful multi-level page table walks and 4KB page protection, by extending the emulator to support Supervisor-mode (S-mode) and implementing hardware-managed translation lookaside buffer (TLB) logic.
- Achieved a full-stack system boot of a standard RISC-V Linux kernel as measured by reaching the BusyBox shell in **<500 million emulated cycles**, by implementing a compliant Platform-Level Interrupt Controller (PLIC) and Supervisor-level trap delegation.

EM Particle Simulator | *C++, SDL2, CMake* — June 2025 – July 2025

- Engineered a high-performance custom 3D renderer in **C++/SDL2**, enabling dynamic mesh visualization under real-time electromagnetic and gravitational forces.
- Facilitated accurate real-time computations for **100+** simultaneous meshes by strengthening simulation stability and precision using fourth-order Runge Kutta Integration solvers.
- Cut draw calls by **75%** and boosted FPS by **50%** through the use of backface culling, Z-buffer-based precomputed visibility, and merge-sort batching to streamline render pipeline.

Digital Circuit Error Analysis | *Verilog, GTKWave, SystemVerilog, Icarus* — February 2025 - April 2025

- Implemented **Verilog** modules for Hamming, CRC, and BCH error-correction codes, automating validation against **1000+** test vectors via scripted **SystemVerilog** testbenches.
- Conducted resource vs. performance trade-off analysis across **30+** block-size configurations, finding **BCH** delivered **90.9%** stronger error correction than **Hamming** at the cost of **3x** greater logic utilization.
- Authored a research report presenting statistical evaluation of algorithmic efficiency and hardware cost, recommending **FPGA** deployment strategies based on error-correction performance versus resource trade-offs.

Technical Skills

Languages: C++/C, Python, Verilog/SystemVerilog, Assembly, Bash, TypeScript/JavaScript, Java, MATLAB, SQL
Hardware: Arduino, FPGAs, Raspberry Pi, ESP32, Breadboarding, Microcontrollers
Frameworks: React, Node.js, Next.js, Flask, Django, FastAPI, TailwindCSS, Bootstrap
Developer Tools: Git, Linux, VS Code, Excel, Vercel, Jupyter Notebook, Amazon Web Services, CMake, GTKWave