Yashwant Dubbaka

248-422-4788 | yrdubbaka@gmail.com | yashwantdubbaka.com | www.linkedin.com/in/yashwant-dubbaka/ | github.com/YD2023 | US Citizen

Education

University of Michigan — Ann Arbor, MI

Dual Degree - B.S.E. Computer Engineering, Bachelor of Business Administration — *Grad Date:* ***May 2027***

Minor in Mathematics — *GPA: 3.67*

- Coursework: VLSI Design, Computer Architecture, Integrated Circuits, Machine Learning, Networking Systems, Digital Design, Computer Organization, Signals and Systems, Data Structures and Algorithms

Technical Skills

Verification: UVM, SVA, Constrained-random Stimulus, Functional Coverage, Formal Property Verification, RAL
Languages/Libraries: SystemVerilog, Verilog, Python, C/C++, TCL, Bash
Tools: Verdi, VCS, JasperGold, ChipStack, Claude Code, QuestaSim, CocoTB, Make, Linux/LSF
Protocols: DDR4/DFI, AMBA AHB, on-chip interconnect (NoC)
Custom Design: Cadence Virtuoso, Spectre, Calibre DRC/LVS, ADE

Experience

Qualcomm — May 2026 - August 2026

Design Verification Engineer Intern — *Santa Clara, CA*

- Architected **UVM** environment for authentication IP from raw RTL, applying **ChipStack** agentic generation to boilerplate; reached regression sign-off for SoC integration in **5 weeks** against team's **15-week** manual estimate
- Authored verification test plan spanning **14 features and 120+ sub-features** of the security architecture, driving an **80+ test** suite covering negative stimulus and illegal-access scenarios
- Closed **code and functional coverage** by mining regressions for uncovered bins and writing targeted **constrained-random** sequences, eliminating every coverage hole except documented waivers for unreachable logic
- Developed **40+ SystemVerilog assertions** for control-path behavior and verified **AHB** protocol compliance with **Cadence ABVIP**, debugging **JasperGold** counterexamples until each targeted property reached full proof
- Owned **bug closure loop** with design teams by isolating regression failure signatures to specific RTL logic and filing reproducible testcases, re-qualifying every RTL drop within **48 hours** to hold the verification schedule

Deloitte — June 2025 – August 2025

Software Engineer Intern — *Detroit, MI*

- Built **Python** ETL pipeline processing **50,000+** threat intelligence records, cutting processing time ∼**60%** by replacing row-wise lookups with parameterized **SQL** existence checks

Projects

Torus 4x4 Mesh NoC | *SystemVerilog, CocoTB, Python, IVerilog* — June 2026 – Present

- Designed **16-router bidirectional torus NoC** in **SystemVerilog** with deterministic routing, **2 virtual channels**, and dateline deadlock avoidance, verifying freedom from deadlock across all traffic patterns
- Built **CocoTB** environment with packet scoreboarding, randomized backpressure, and functional coverage, closing **57 tests** over **4,608 packets** and all valid endpoint pairs
- Characterized throughput and latency across **156 simulations** sweeping FIFO depth, offered load, and traffic pattern, identifying **depth-4** buffering as the saturation knee

RISC-V 16-bit Processor | *Cadence Virtuoso, APR, Verilog, Spectre, Calibre, Layout* — January 2026 – May 2026

- Designed 16-bit RISC-V processor integrating decoder, register file, ALU, shifter, memory, and branch-capable PC
- Implemented radix-2 **Kogge-Stone** ALU adder in **Cadence**, reducing carry propagation from **O(N) to O(logN)**
- Optimized sizing and speed paths with Spectre simulations, reducing ALU rise delay from **467ps to 132ps**
- Completed CMOS physical layout with port planning, metal routing, and Calibre DRC/LVS verification, reducing adder area by **32%** while managing interconnect length, capacitance, and high-fanout routing constraints

DDR4 Memory Subsystem | *SystemVerilog, UVM, QuestaSim, DFI* — May 2025 – August 2025

- Architected **UVM** environment for **DDR4** memory controller with **dual-agent** architecture and scoreboard, validating data integrity across **65,000+** memory locations
- Implemented **SystemVerilog** assertions and reference model checking **DFI** protocol timing across a **6-state** controller FSM, closing functional coverage over **10+** scenarios