Randy Widialaksono

ASIC Design/Software Engineer https://codify.dev • email: r[at]codify.dev

Objective & Summary

Advancing ASIC design quality and methodology by building novel, performant tools, harnesses, and models. Electrical engineering background in computer architecture, ASIC design, ML, and VLSI physical design automation. Recent experience: developing a core C++ CAD library and tools for early floorplanning and shift-left initiatives. Drove end-to-end RTL-to-GDS2 3D-IC tape-out producing functional silicon. Interests: RL, agentic flows, Rust.

Experience

NVIDIA — Austin, Texas

VLSI CAD Architect and Developer / Software R&D Engineer – ASIC Physical Design *2020–present*

Developed, maintained, and deployed in-house C++ library/tools suite used by design and methodology teams across the company. Managed and optimized build processes, dependency management, Python/Rust binding generation, regression testing infrastructure, cross-tool integration, and deployment troubleshooting.

Eliminated production-flow bottlenecks in chip integration and physical design by building high-performance tools that cut manual work or slash runtime, typically from hours to minutes. Built design flows to improve productivity and quality by conducting research, collaborating across teams, and implementing algorithms.

Applied data-oriented design and user-centric API design preferring standard data exchange formats.

Explored applications of agentic AI/LLM: developing a custom harness and applying a zero-token architecture.

Sample scope: top-level reuse-aware bus route planning, feedthrough routing heuristics, floorplan graph extraction & traversal, ACO, spanning trees, GUI development, multithreading, chiplet RTL build, ECO, netlist/layout query & editing, readers/writers, computational geometry, pinning, constraint generation, and custom routing.

Intel — Austin, Texas

Silicon Architecture Engineer – SoC Performance Architect *2017–2020*

Integrated x86 model into a C++ system-level simulator. Applied machine learning for performance projections.

Design Automation Engineer – Physical Design *2016–2017*

Developed, configured, and deployed EDA-based in-house physical design flows for partition/block-level closure.

North Carolina State University (NCSU) — Raleigh, North Carolina

Graduate Teaching Assistant & Research Assistant *2010–2016*

Led design and implementation of a 3D-IC processor fabricated on IBM/GF + Tezzaron TSV die-stacking process.

Developed and executed RTL-to-GDSII flow: synthesis, floorplanning, P&R, timing closure, DRC & LVS fixes.

Qualcomm Research — San Diego, California

Research Intern: mixed-signal circuit design verification, FPGA prototyping. *Summer 2013*

Skills

Programming	VLSI Implementation	Dev Tools
C++, **Python**, Tcl.	SystemVerilog/Verilog, SystemC.	CMake, Pybind11, CI/CD, Ninja.
Basic: Java, Perl, SQLite, R,	ICC2, Innovus, PrimeTime, Calibre.	Git, Perforce, Qt, ImGUI, PyTorch.
HTML, JS / TypeScript, Rust.	Place-Route, DFT, Timing, DRC/LVS.	Marimo, Pandas, Coding Harnesses.

Select Publications/Awards

A Logic-on-logic 3D-stacked Heterogeneous Multi-core Processor. IEEE ICCD Conference 2017.

Physical Design of a 3D-stacked Heterogeneous Multi-core Processor. IEEE 3D-IC Conference 2016.

Reuse-Based Optimization for Pre-Bond and Post-Bond Testing of 3D-Stacked ICs. IEEE Transactions on CAD vol. 34.

Ranked 34th in USA, IEEExtreme 24-hour Programming Competition, 2014. Team of 2.

Best FPGA Implementation at International LSI Design Contest, Japan 2009. Xilinx Award. Team of 3.

Education

North Carolina State University — Raleigh, North Carolina

Ph.D. in Computer Engineering *3.98/4.0*

Dissertation: Three-Dimensional Integration of Heterogeneous Multi-Core Processors.

Built automated 3D-IC flow optimizing paths across 3D vias; drove architectural feedback, RTL verification, and GDSII signoff. Taped out 3 chips on MPW runs. **TA**: Design of Digital Systems, Computer Design & Tech.

Graduate Coursework

Software Engineering	Advanced Microarchitecture	ASIC Design	Physical Design
Memory Systems	Parallel Computer Arch.	ASIC Verification	Electronic System Level Design
Computer Networks	Computer Design & Tech.	IC Technology & Fabrication	VLSI Systems Design
Embedded Systems Design	Digital Electronics	Modern Computer Algebra-AU	VLSI System Testing (Duke U.)

Duke University — Durham, North Carolina

Visiting Scholar: coursework, research collaboration on DFT and 3D-IC CAD. *2013*

Bandung Institute of Technology — West Java, Indonesia

B.S. in Electrical Engineering *2009*

Thesis: C implementation of neural networks and Kohonen SOM (training/inference, floating/fixed-point) on a multi-core Parallax microcontroller. Teaching Assistant: Digital Systems, Microprocessor Lab.

Oita University — Oita, Japan

Research Exchange Student: Reinforcement Learning Lab (Prof. Shibata), with coursework *2007–2008*

Implemented neural networks in C for face following on a panning camera and reinforcement learning for hardware control.

Project Experience (Academic/Side)

End-to-end Silicon Implementation / Tape-outs

Successful academic tape-out (functional fabricated 3D-IC processor chip) of a heterogeneous multi-core processor system with thread migration features at NCSU. Developed visualizations and fully automated back-end flow. Performed final physical verification (DRC/LVS) and layout fixes (e.g. antenna rule violations) for signoff.
Silicon implementation has two stacked dies of 5.25 mm $\times$ 5.25 mm on a 130 nm process.

RTL Design (Verilog), FPGA Prototyping, bare-metal programming (C/asm)

Implemented Sokoban (moving box puzzle game) on FPGA: coded the game and sprites in C and MIPS assembly from scratch. Wrote pipelined MIPS processor in Verilog RTL with data forwarding and a Kogge-Stone adder. Wrote Verilog code for debouncing FPGA buttons and rendering VGA display.

Machine Learning (PyTorch)

Implemented quantized LeNet with feedback alignment and custom training algorithms for quantized NNs.

Memory Systems

Performed modeling and performance comparison between ideal and non-ideal block placement policies for multi-core systems. Cache block placement policy: requestor core cache vs remote core cache. Analyzed experiment results from running SPEC2K benchmarks in SIMICS.

ESL & Physical Design (SystemC, HLS)

Performed TLM & ESL modeling of an SoC design comprising an ARM Cortex core, DRAM model, and AMBA bus. Performed physical design optimizations, signal integrity/power/timing analysis. Tools: SystemC, Mentor Graphics Vista, Catapult, Python, C++, UML, Encounter, PrimeTime.

Parallel Computer Architecture

Implemented a cache coherence simulator in C++ and protocol enhancements to reduce off-chip memory accesses.

Computer Design and Technology (C++)

Implemented cache, branch target buffer, and Tomasulo superscalar processor simulators.
Implemented a checkpoint recovery mechanism for a large fetch window processor within SimpleScalar simulator.

Advanced Microarchitecture (C++)

Implemented and compared thread migration (across cores) strategies within SimpleScalar simulator framework.

ASIC Verification (SystemVerilog)

Verified an out-of-order superscalar core for tape-out, uncovering design bugs in the load-store unit and issue queue.

Digital Electronics (CMOS circuit design)

Designed a low-power Hybrid Latch Flip-flop in academic 45 nm tech library. Operating clock frequency 4 GHz, power consumption 19.9 μW, setup time 13.5 ps, hold time 86 ps, t_{DQ} of 63.64 ps.
Designed a voltage-mode and current-mode differential transmitter circuit. Tools: HSPICE.

VLSI Systems Design (logic design, physical layout)

Designed a full-custom 3×3 arbiter-crossbar CMOS unit, 2nd best performance and energy-delay-squared metric out of 27 teams. Customized power delivery network and clock tree design. Created custom standard cell library and top-level integration. Achieved 5.5 GHz clock frequency, 0.19 nW power, with FreePDK45 technology library. Tools: Cadence Virtuoso, HSPICE, Calibre DRC-LFD.

ASIC Design (Verilog)

Implemented a Viterbi Decoder in RTL Verilog. Optimized throughput and delay per unit area metric by designing a fast floating-point unit, using dual-port memory, and pipelining.