Research Experience for Undergraduates (REU)

Summer Projects 2026

Section A: Hardware-focused projects

Projects involving circuit design, physical fabrication, lab measurement, or chip submission

Project 1: A hardware accelerator for a spiking neural network from algorithm to silicon

Research question

Can a complete hardware design flow -- from a trained spiking neural network (SNN) model to a submitted application-specific integrated circuit (ASIC) layout in a standard complementary metal-oxide semiconductor (CMOS) process -- be executed within a ten-week undergraduate research cycle using open-source electronic design automation (EDA) tools, and what are the dominant resource bottlenecks at each abstraction layer?

Overview

The intern designs a small SNN inference accelerator, from a PyTorch model to a fabricated or submitted chip through Tiny Tapeout using the SkyWater Sky130 CMOS process design kit (PDK). The goal is a complete vertical slice through the semiconductor hardware design stack.

Semiconductor relevance

Synthesis, place-and-route, and physical verification target the Sky130 130 nm bulk CMOS PDK. Area, power, and timing results are reported in standard CMOS figures of merit (gates per mm², dynamic power per inference, critical path delay), giving the intern direct experience with the metrics that drive real semiconductor product decisions.

Weekly plan

Weeks 1-2	Train a small SNN on N-MNIST or CIFAR-10 (Canadian Institute for Advanced Research 10-class image dataset) using snnTorch [1]. Quantize weights to 4-bit fixed point. Characterize multiply-accumulate (MAC) and memory footprint with torchinfo and thop.
Weeks 3-4	Translate the inference graph into synthesizable Verilog (leaky integrate-and-fire (LIF) neuron unit, systolic-array weight buffer). Verify functionally using cocotb [2] co-simulated against the snnTorch model.
Weeks 5-6	Run synthesis, place-and-route, and timing closure with OpenLane 2 / Sky130 PDK. Prepare Tiny Tapeout submission [3]. Feed area and power reports back to algorithm-layer tradeoffs.
Weeks 7-8	Run gate-level simulation with cocotb. Sign off design rule check (DRC) / layout versus schematic (LVS). Complete Tiny Tapeout submission with a design document recording all bit-width and cell utilization tradeoffs.
Weeks 9-10	Characterize the chip on a breakout board if returned; otherwise evaluate in simulation. Write a short technical report structured as a conference paper.

Deliverables

Trained SNN model, register-transfer level (RTL) source, cocotb testbench suite, OpenLane configuration, Tiny Tapeout submission, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Eshraghian, J. K., et al. (2023). Training spiking neural networks using lessons from deep learning. Proceedings of the IEEE, 111(9), 1016-1054.

[2] Rosser, B., et al. (2018). cocotb: A Python-based hardware verification framework. github.com/cocotb/cocotb

[3] Tiny Tapeout project documentation. tinytapeout.com

Project 2: Physical computation in resistive networks

Research question

Under what conditions does a passive resistive or memristive network solve combinatorial optimization problems through physical relaxation, and how does resistive random-access memory (RRAM) memristive state alter the energy landscape relative to an ideal resistor network?

Overview

This project investigates substrate-native computation, asking whether a passive memristive network can solve optimization problems by relaxing to a physical equilibrium. Device models are parameterized for complementary metal-oxide semiconductor (CMOS)-compatible HfOx RRAM, and simulation program with integrated circuit emphasis (SPICE) simulations use process-accurate models throughout.

Semiconductor relevance

Devices are modeled after hafnium oxide (HfOx) RRAM, a leading candidate for monolithic integration at 28 nm and below [3]. The SPICE environment uses Berkeley short-channel insulated-gate field-effect transistor model 4 (BSIM4) metal-oxide-semiconductor field-effect transistor (MOSFET) models for selector transistors. The breadboard prototype is a proof-of-concept for a circuit ultimately realizable as a mixed-signal application-specific integrated circuit (ASIC).

Weekly plan

Weeks 1-2	Study Hopfield's energy formulation [1] and analog Ising machine literature [2]. Implement a software Hopfield network and verify it solves small maximum cut (MAX-CUT) instances.
Weeks 3-4	Map the Hopfield energy function to a passive resistor network following analog very-large-scale integration (VLSI) conductance-weight correspondence [4]. Simulate in ngspice with ideal resistors and BSIM4 selectors.
Weeks 5-6	Replace ideal resistors with threshold-type empirical additive memristance (TEAM)-model HfOx RRAM memristors. Characterize how dynamic resistance state alters the energy landscape and whether convergence is reliable.
Weeks 7-8	Build a 4-8 node breadboard prototype (KiCad printed circuit board (PCB)). Probe with a data acquisition (DAQ) board and compare measurements against SPICE predictions.
Weeks 9-10	Quantify simulation-vs-physical divergence. Discuss scalability to CMOS integration. Write the technical report and contribute to Summer Proceedings 2026.

Deliverables

SPICE netlists (BSIM4 models), KiCad design files, measurement data, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences (PNAS), 79(8), 2554-2558.

[2] Mohseni, N., et al. (2022). Ising machines as hardware solvers of combinatorial optimization problems. Nature Reviews Physics, 4(6), 363-379.

[3] Prezioso, M., et al. (2015). Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature, 521, 61-64.

[4] Mead, C. (1990). Neuromorphic electronic systems. Proceedings of the IEEE, 78(10), 1629-1636.

Project 3: Fractional-order neuromorphic hardware -- from model to circuit

Research question

Can fractional-order leaky integrate-and-fire (FO-LIF) neuron dynamics, known to capture multi-timescale spike adaptation, be realized in analog complementary metal-oxide semiconductor (CMOS) circuits at a complexity and power envelope compatible with neuromorphic hardware integration?

Overview

Fractional-order differential equations capture history-dependent neuronal dynamics that integer-order models cannot reproduce without many auxiliary state variables [1]. This project bridges the mathematical formalism and CMOS circuit implementation, targeting the Sky130 process design kit (PDK) for simulation program with integrated circuit emphasis (SPICE) verification.

Semiconductor relevance

The circuit targets standard analog CMOS primitives in the Sky130 PDK: differential pairs, current mirrors, and transconductance-capacitor (Gm-C) integrators. The fractional-order operator maps to a passive resistor-capacitor (RC) ladder realizable in CMOS back-end-of-line (BEOL) metal layers. Power and area estimates use Sky130 device parameters, making results directly comparable to published CMOS neuromorphic designs [4].

Weekly plan

Weeks 1-2	Review the FO-LIF model and its behavioral differences from the standard LIF model, focusing on sub-threshold adaptation and power-law inter-spike intervals. Simulate with fde-code or FractionalDiffEq.jl.
Weeks 3-4	Survey fractional-order approximation methods (Oustaloup recursive filter, Charef pole-zero expansion, continued fraction expansion). Evaluate accuracy vs. circuit complexity tradeoffs [2].
Weeks 5-6	Design an analog CMOS FO-LIF circuit in SPICE with Sky130 device models (Gm-C integrator + fractional approximation network). Verify spike trains match the mathematical model.
Weeks 7-8	Substitute the RC approximation with a SPICE memcapacitor model [3]. Compare accuracy and simulated power against the all-CMOS baseline.
Weeks 9-10	Benchmark the FO-LIF circuit on a rate adaptation task. Compare energy and accuracy against an integer-order CMOS baseline. Write the technical report.

Deliverables

FO-LIF simulation code, SPICE netlists (Sky130 device models), benchmarking results, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Teka, W., et al. (2017). Spiking and bursting patterns of fractional-order Izhikevich model. Communications in Nonlinear Science and Numerical Simulation (CNSNS), 56, 161-176.

[2] Valsa, J., & Vlach, J. (2013). RC models of a constant phase element. International Journal of Circuit Theory and Applications, 41(1), 59-67.

[3] Biolek, D., et al. (2010). SPICE model of memcapacitor. Electronics Letters, 46(7), 520-522.

[4] Mead, C. (1990). Neuromorphic electronic systems. Proceedings of the IEEE, 78(10), 1629-1636.

Project 4: Honey as a fractional-order reservoir computing substrate

Research question

Does a copper-electrode crossbar on a semi-fluid honey film behave as a network of fracmemristive constant-phase elements (CPEs), can the fractional-order exponent alpha be tuned through temperature and humidity, and can the substrate perform the nonlinear autoregressive moving-average order 10 (NARMA-10) benchmark at a level comparable to simulated integer-order reservoirs of the same dimensionality?

Overview

Honey is a viscoelastic material whose rheological and electrical impedance properties are described by fractional-order differential equations [1,2]. When contacted by copper electrodes it forms an electrochemical interface analogous to resistive random-access memory (RRAM) [3]. This project builds and characterizes a low-cost fractional-order reservoir computer and evaluates it on NARMA-10.

Semiconductor relevance

The copper/honey/copper oxide (CuxO) interface is directly analogous to the Cu/oxide/Cu structure of conductive-bridge random-access memory (CBRAM), a class of complementary metal-oxide semiconductor (CMOS)-integrated non-volatile memory [3]. The impedance spectroscopy protocol -- CPE fitting across 1 Hz to 1 MHz -- is the same methodology used for RRAM reliability studies and high-k gate dielectric characterization.

Weekly plan

Weeks 1-2	Fabricate a 3×3 copper tape crossbar on a glass slide. Deposit semi-fluid honey at each junction. Seal in a humidity-controlled chamber (sodium chloride (NaCl), ~75% relative humidity (RH)). Collect baseline impedance spectra at 25°C; fit CPE parameters (alpha, Q).
Weeks 3-4	Repeat impedance spectroscopy at 20-40°C in 5°C steps. Plot alpha(T). A decreasing alpha with decreasing temperature is predicted by rheology [1,2] and tested electrically here.
Weeks 5-6	Verify the echo state (fading memory) property using a standard wash-in protocol. Monitor impedance drift per session; discard sessions where alpha shifts by more than 0.05.
Weeks 7-8	Run waveform classification (sine, square, sawtooth) as a sanity check. Then run NARMA-10 with ridge regression readout. Report normalized root-mean-square error (NRMSE) and compare against published echo state network results [4].
Weeks 9-10	Repeat NARMA-10 with a glycerol-water control substrate (matched viscosity, no viscoelasticity). Compare performance to test whether fractional-order dynamics contribute measurably.

Deliverables

Impedance spectroscopy dataset (alpha vs. temperature), CPE fitting code, NARMA-10 results for honey and glycerol-water control, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Oroian, M. (2012). A viscoelastic model for honeys using the time-temperature superposition principle. Food and Bioprocess Technology, 6, 2217-2228.

[2] Gasparoux, J., et al. (2008). Large frequency bands viscoelastic properties of honey. Journal of Non-Newtonian Fluid Mechanics, 153, 46-52.

[3] Sueoka, B., & Zhao, F. (2022). Memristive synaptic device based on honey. Journal of Physics D: Applied Physics, 55, 225105.

[4] Tanaka, G., et al. (2019). Recent advances in physical reservoir computing. Neural Networks, 115, 100-123.

Project 5: Can a large language model propose analog amplifier topologies that are structurally outside its training distribution?

Research question

When a large language model (LLM) is used in a simulation-in-the-loop framework to iteratively propose and refine analog amplifier topologies in response to simulation program with integrated circuit emphasis (SPICE) feedback, do the topologies it generates remain within the distribution of circuits seen in training data, or does iterative feedback drive it toward structurally novel configurations -- and can novelty be measured independently of performance?

Overview

Analog circuit synthesis has no automated equivalent of logic synthesis: there is no established path from a behavioral specification to a transistor-level schematic [1]. Recent work has begun exploring LLMs as topology generators [2], but it is not known whether LLM-generated topologies are genuinely novel or interpolations of training examples. This project implements a SPICE-in-the-loop LLM design loop for a two-stage operational amplifier (op-amp) specification and measures topological novelty of the outputs.

Semiconductor relevance

All generated topologies are simulated using ngspice with Sky130 process design kit (PDK) device models, ensuring proposed circuits are evaluated under realistic complementary metal-oxide semiconductor (CMOS) process constraints. Novelty is measured by comparing generated netlists against the Analog Design Automation benchmark corpus [3] using graph edit distance (GED) on the transistor-level circuit graph.

Weekly plan

Weeks 1-2	Define the target specification: a two-stage CMOS op-amp with gain-bandwidth product (GBW) > 10 MHz, phase margin (PM) > 60°, and quiescent current < 100 µA in Sky130. Collect 20-30 reference topologies from textbooks and open repositories to form the novelty baseline corpus [3].
Weeks 3-4	Implement the LLM design loop: prompt an LLM (via API) with the specification and a SPICE netlist template; parse the returned netlist; run ngspice; extract GBW, PM, and power; format results as feedback for the next iteration.
Weeks 5-6	Run 50-100 design iterations. Log every proposed topology as a circuit graph (nodes = transistors, edges = nets). Compute GED from each generated topology to the nearest neighbor in the reference corpus.
Weeks 7-8	Analyze the relationship between novelty (GED from corpus) and performance (GBW, PM). Test whether the LLM converges to low-novelty, high-performance solutions or whether high-novelty topologies also appear on the Pareto frontier.
Weeks 9-10	Select the highest-novelty topology that meets specification. Perform a manual design review to determine whether novelty is meaningful (different feedback path, novel bias configuration) or superficial (equivalent circuit with relabeled nodes). Write the technical report.

Deliverables

LLM design loop codebase, generated topology corpus (50-100 netlists with SPICE results), novelty measurement scripts (GED), Pareto frontier analysis, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Settaluri, K., et al. (2020). AutoCkt: Deep reinforcement learning of analog circuit designs. Design, Automation & Test in Europe (DATE), 490-495.

[2] Chang, T.-Y., et al. (2024). ChipNeMo: Domain-adapted LLMs for chip design. arXiv:2311.00176.

[3] Kunal, K., et al. (2020). ALIGN: Open-source analog layout automation from the ground up. IEEE Journal of Solid-State Circuits (JSSC), 56(1), 60-71.

[4] Lyu, W., et al. (2018). An efficient Bayesian optimization approach for automated optimization of analog circuits. IEEE TCAS-I, 65(6), 1954-1967.

Project 6: How does conductance variability in a VO_2 oscillator array propagate to ordinary differential equation solution error, and can a compiler correct it?

Grant

Schmidt Sciences Unconventional Compute RFP 2026 -- VO_2 Mott Analog Computer

Research question

In a simulated vanadium dioxide (VO_2) oscillator network where coupling conductances are subject to device-to-device variability drawn from experimentally reported distributions, how does ordinary differential equation (ODE) solution error scale with network size and variability magnitude -- and can a compiler-level calibration step that adjusts programmed conductance values to compensate for measured offsets reduce error to below the 10% threshold required for useful analog computation?

Overview

The Schmidt Sciences VO_2 proposal maps ODE coupling coefficients to programmable $HxVO_2$ conductances (G_ij), but device variability of 5-10% is reported for these elements. Before hardware experiments can be interpreted, the relationship between conductance error and solution error must be understood quantitatively. This project studies that relationship in simulation for the IEEE 9-bus power system swing equations and proposes a compiler calibration strategy.

Semiconductor relevance

VO_2 is a correlated-electron Mott semiconductor deposited by radio-frequency (RF) magnetron sputtering onto complementary metal-oxide semiconductor (CMOS)-compatible substrates. The coupling conductances (G_ij) are hydrogen-doped VO_2 ($HxVO_2$) elements programmed electrically. This project models the full conductance-to-solution error chain that the hardware team must characterize at the go/no-go gate at month 9 of the Schmidt proposal, providing a predictive simulation baseline before the 16-oscillator array is built.

Weekly plan

Weeks 1-2	Implement the IEEE 9-bus power grid swing equations [1] as a coupled ordinary differential equation (ODE) system in Python/SciPy. Verify against published reference trajectories for the standard N-1 contingency set. This is the ground-truth digital solver.
Weeks 3-4	Build a lumped-parameter model of the VO_2 oscillator network: each oscillator is a van der Pol relaxation oscillator [2] whose rest frequency and coupling strength are set by conductance values. Implement Monte Carlo conductance sampling from a Gaussian distribution with standard deviation sigma = 0.05 and 0.10 of the nominal value.
Weeks 5-6	Run 500 Monte Carlo trials per sigma level for the 9-bus system. Compute solution error as the normalized root-mean-square error (NRMSE) between the analog oscillator trajectory and the reference digital trajectory. Plot error vs. sigma and error vs. network size (9, 14, 39 nodes).
Weeks 7-8	Implement a compiler calibration strategy: given a known conductance error map (simulating a post-fabrication measurement step), adjust programmed G_ij values to compensate. Measure the reduction in NRMSE achieved by calibration for each sigma level. Identify the sigma threshold above which calibration fails to recover acceptable accuracy.
Weeks 9-10	Produce a go/no-go decision support document: for the variability levels reported experimentally (5-10%), does simulation predict that calibration can achieve >90% stability classification accuracy on the 9-bus contingency set? Write the technical report.

Deliverables

Monte Carlo simulation framework (Python), NRMSE vs. variability plots, calibration algorithm, go/no-go support document for hardware team, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Kundur, P. (1994). Power System Stability and Control. McGraw-Hill.

[2] Dutta, S., et al. (2021). An Ising Hamiltonian solver based on coupled stochastic phase-transition nano-oscillators. Nature Electronics, 4, 502-512.

[3] Deng, S., et al. (2023). Selective area doping for Mott neuromorphic electronics. Science Advances, 9(11), eade4838.

Project 7: At what device pitch does thermal cross-talk in a VO_2 oscillator array exceed the stability budget required for useful analog computation?

Grant

Schmidt Sciences Unconventional Compute RFP 2026 -- VO_2 Mott Analog Computer

Research question

In a two-dimensional (2D) array of vanadium dioxide (VO_2) relaxation oscillators driven at sustained current levels, at what minimum device pitch does predicted steady-state thermal cross-talk between adjacent oscillators exceed the 5% per hour frequency drift budget specified in the proposal go/no-go criteria -- and how does that critical pitch scale with drive current and substrate thermal conductivity?

Overview

The Schmidt Sciences proposal identifies thermal cross-talk between adjacent oscillators as an open risk and specifies a go/no-go criterion of <5% frequency drift per hour under load. No quantitative prediction of the critical device pitch exists. This project builds a lumped-parameter and finite-element thermal model of the array geometry and maps the phase diagram of safe vs. unsafe operating conditions before the hardware team commits to a chip layout.

Semiconductor relevance

VO_2 undergoes an insulator-to-metal transition (IMT) at approximately 68°C, making its oscillation frequency strongly sensitive to local temperature. The device geometry -- thin-film VO_2 on a SiO_2/Si or sapphire substrate with metal electrodes -- is standard for Mott device fabrication. The thermal model uses published values for VO_2, SiO_2, and Si thermal conductivity, and predicts cross-talk as a function of pitch, drive current, and substrate choice -- directly actionable parameters for the Rutgers fabrication team.

Weekly plan

Weeks 1-2	Review published VO_2 device geometry and thermal properties [1,2]. Build a 1D lumped-parameter thermal model of two adjacent oscillators on a shared substrate: each device is a heat source; thermal resistance between devices is a function of pitch and substrate conductivity.
Weeks 3-4	Extend to a 2D finite-difference thermal model of an 4×4 array. Parameterize by pitch (5-100 µm), drive current (0.5-5 mA), and substrate (SiO_2/Si vs. sapphire). Validate the model against any published cross-talk measurements in VO_2 or NbO_2 oscillator arrays [2].
Weeks 5-6	Run parametric sweeps. For each (pitch, current, substrate) combination, compute the steady-state temperature rise at each oscillator and convert to predicted frequency shift using the published VO_2 frequency-temperature coefficient [1]. Identify the critical pitch at each operating point.
Weeks 7-8	Plot the phase diagram of safe vs. unsafe regions in (pitch, current) space for each substrate. Identify the operating envelope that satisfies the <5% drift criterion. Determine whether the pitch constraints are compatible with the wafer-scale fabrication process (minimum feature size, electrode geometry) described in the proposal.
Weeks 9-10	Deliver a layout recommendation to the Rutgers fabrication team: minimum safe pitch for each substrate and current level, with quantified margin. Write the technical report.

Deliverables

Lumped-parameter and finite-difference thermal models (Python), phase diagram of safe operating conditions, layout recommendation document for fabrication team, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Yi, W., et al. (2018). Biological plausibility and stochasticity in scalable VO_2 active memristor neurons. Nature Communications, 9, 4661.

[2] Shukla, N., et al. (2014). Synchronized charge oscillations in correlated electron systems. Scientific Reports, 4, 4964.

[3] Deng, S., et al. (2023). Selective area doping for Mott neuromorphic electronics. Science Advances, 9(11), eade4838.

Project 8: What is the minimum analog solution accuracy required for correct transient stability classification in power grid contingency analysis?

Grant

Schmidt Sciences Unconventional Compute RFP 2026 -- VO_2 Mott Analog Computer

Research question

For the IEEE 9-bus, 14-bus, and 39-bus power grid transient stability benchmarks, what is the minimum trajectory accuracy -- measured as normalized root-mean-square error (NRMSE) relative to a reference digital integrator -- at which an analog computer's output still yields correct stability classification, and does that accuracy threshold vary systematically with contingency type (fault location, fault duration, generator trip)?

Overview

The Schmidt Sciences proposal claims that the VO_2 analog computer can classify power grid stability without full trajectory precision. This claim has not been quantified. Before hardware is built and benchmarked, it is necessary to know how much trajectory error is tolerable for the classification task. This project establishes that threshold using a software simulation of the swing equations, injecting controlled amounts of noise into the solution trajectory and measuring when classification decisions flip.

Semiconductor relevance

This is a pre-hardware characterization study that directly supports the Schmidt proposal go/no-go gate at month 9, which requires >90% classification accuracy. The study uses only software (Python, SciPy), but its output -- a quantitative accuracy-classification threshold map -- is the primary metric the hardware team will use to interpret their first experimental results on the 8-16 oscillator array. The study is also relevant to any analog computing platform (VO_2, NbO_2, memristive) targeting power systems applications.

Weekly plan

Weeks 1-2	Implement the IEEE 9-bus, 14-bus, and 39-bus swing equation systems in Python/SciPy [1]. Generate a reference contingency library: 50 fault scenarios per test system (N-1 line outages, generator trips, fault-clear sequences), each labeled stable or unstable by the reference digital integrator (Runge-Kutta 4/5).
Weeks 3-4	Implement a noise injection model: add Gaussian noise with standard deviation sigma to each trajectory state variable, where sigma is expressed as a fraction of the trajectory range. Test sigma values from 0.01 to 0.30 (1% to 30% trajectory noise). For each sigma level, re-classify each contingency using the noisy trajectory and record whether the classification matches the reference label.
Weeks 5-6	Plot classification accuracy vs. sigma for each test system and contingency type (fault location, fault duration, generator trip). Identify the sigma threshold at which accuracy drops below 90% (the go/no-go criterion). Test whether fault-on-vs-clear-from-bus contingencies are harder to classify than generator-trip contingencies at matched noise levels.
Weeks 7-8	Investigate whether the accuracy threshold depends on the noise structure (white noise vs. systematic bias vs. correlated oscillator noise). Implement a simple majority-vote classifier that aggregates three independent noisy trajectories and measure whether ensemble averaging recovers accuracy above threshold.
Weeks 9-10	Deliver a quantitative accuracy requirement document: for each test system, the minimum NRMSE below which the VO_2 analog computer must operate to meet the 90% classification criterion. Write the technical report.

Deliverables

Contingency library (Python, 150+ labeled scenarios), noise injection simulation, accuracy-vs-sigma plots, accuracy requirement document for hardware team, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Kundur, P. (1994). Power System Stability and Control. McGraw-Hill.

[2] Milano, F. (2005). An open source power system analysis toolbox. IEEE Transactions on Power Systems, 20(3), 1199-1206.

[3] Aristidou, P., et al. (2014). Dynamic simulation of large-scale power systems using a parallel Schur-complement-based decomposition method. IEEE TPDS, 25(10), 2561-2570.

Section B: Software-focused projects

Projects involving tool development, simulation frameworks, ML modeling, and evaluation

Project 9: A co-design tool for fractional-order neuromorphic circuits

Research question

What abstractions and automated design-space exploration strategies are necessary for a software tool that co-optimizes the mathematical order, circuit approximation topology, and complementary metal-oxide semiconductor (CMOS) device parameters of a fractional-order neuromorphic circuit against user-specified accuracy and power constraints?

Overview

No general-purpose co-design tool exists for fractional-order neuromorphic circuits targeting CMOS. Designers currently iterate manually across model order, rational approximation, and circuit implementation. This project produces a Python-based tool that automates this pipeline with simulation program with integrated circuit emphasis (SPICE) back-end support.

Semiconductor relevance

The netlist generator targets the Sky130 process design kit (PDK) and, via a configuration switch, commercial foundry PDKs (Taiwan Semiconductor Manufacturing Company (TSMC), GlobalFoundries) through standard SPICE formats. Device parameter ranges are bounded by foundry-specified limits for CMOS back-end-of-line (BEOL) passives. The scoring function reports estimated die area, dynamic power from transistor-level simulation, and compatibility with standard analog synthesis flows.

Weekly plan

Weeks 1-2	Survey existing electronic design automation (EDA) tools for analog neuromorphic design. Document manual design decisions the tool must automate. Define the tool's input/output specification.
Weeks 3-4	Implement Oustaloup, Charef, and continued-fraction-expansion approximation methods in Python [1]. Unified interface: order alpha + frequency range -> rational transfer function. Validate against published benchmarks.
Weeks 5-6	Write a netlist generator mapping transfer functions to passive resistor-capacitor (RC) ladders compatible with Sky130 BEOL constraints. Invoke ngspice automatically to run transient simulation.
Weeks 7-8	Add a sweep-and-evaluate layer: explore approximation orders and topologies in batch SPICE, parse waveforms, score against user-defined metrics (timing error, die area, power).
Weeks 9-10	Validate on the FO-LIF (fractional-order leaky integrate-and-fire) circuit from Project 3 and one additional literature benchmark. Write user documentation and contribute to Summer Proceedings 2026.

Deliverables

Open-source Python package (MIT-licensed) with command-line interface (CLI) and application programming interface (API), SPICE integration scripts, validation results, user documentation, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Valsa, J., & Vlach, J. (2013). RC models of a constant phase element. International Journal of Circuit Theory and Applications, 41(1), 59-67.

[2] Tepljakov, A., et al. (2018). Towards industrialization of fractional-order proportional-integral-derivative (FOPID) controllers. IEEE Access, 6, 3454-3462.

Project 10: A benchmark suite for memristive and memcapacitive computing systems

Research question

What workload characteristics -- sparsity, temporal structure, precision requirements, and problem scale -- most strongly determine relative performance between memristive and memcapacitive substrates on complementary metal-oxide semiconductor (CMOS), and can a systematic benchmark suite expose these differences while reporting results in International Solid-State Circuits Conference (ISSCC) figures of merit?

Overview

The memristive computing field lacks standardized benchmarks that control for workload characteristics, expose device-type differences, and report results comparable to conventional CMOS hardware. This project designs, implements, and releases a benchmark suite targeting memristive and memcapacitive crossbar architectures on CMOS substrates.

Semiconductor relevance

All runs report ISSCC-standard figures of merit: energy per synaptic operation (pJ/SOP), throughput per silicon area (giga-operations per second per square millimeter (GOPs/mm²)), and normalized accuracy loss. A CMOS-only digital crossbar with static random-access memory (SRAM) weight storage serves as the baseline. Variability distributions are drawn from reported data for hafnium oxide (HfOx) resistive random-access memory (RRAM) and ferroelectric capacitor (FeCAP) devices on 28 nm CMOS [3].

Weekly plan

Weeks 1-2	Review NeuroBench [1], MLPerf Tiny, and ISSCC figures of merit. Identify gaps specific to CMOS-integrated memristive and memcapacitive systems. Produce a gap analysis document.
Weeks 3-4	Select 5-8 benchmark tasks: sparse matrix-vector multiply, temporal pattern classification, reservoir state capacity, energy-per-inference. Specify fixed datasets, topologies, and CMOS figure-of-merit (FoM) metrics.
Weeks 5-6	Integrate benchmark tasks with the crossbar simulation framework from the co-design projects. Run under memristive, memcapacitive, and CMOS-SRAM device models. Collect standardized result files.
Weeks 7-8	Systematically vary workload parameters (sparsity, sequence length, precision) and device parameters (variability, endurance). Measure effect on benchmark scores per device type.
Weeks 9-10	Package benchmark suite, datasets, and analysis scripts for public release. Write the technical report and contribute to Summer Proceedings 2026.

Deliverables

Open-source benchmark suite (MIT-licensed), datasets, analysis scripts, gap analysis document, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Yik, J., et al. (2025). NeuroBench: A framework for benchmarking neuromorphic computing algorithms and systems. Nature Machine Intelligence, 7, 85-99.

[2] Kvatinsky, S., et al. (2013). TEAM: Threshold adaptive memristor model. IEEE TCAS-I, 60(1), 211-221.

[3] Prezioso, M., et al. (2015). Nature, 521, 61-64.

Project 11: Evaluating large language models for high-level synthesis annotation

FAST thrust

FAST (Frontier Automation for Semiconductor Technology) §C.2.a.1 -- INSTA-HLS (Interpretable Smart Toolflow for Accelerators: High-Level Synthesis)

Research question

To what extent do current large language models (LLMs) produce accurate, consistent, and engineer-useful natural language annotations of high-level synthesis (HLS)-generated Verilog, and what hardware constructs -- pipeline stages, resource sharing, loop unrolling -- are most frequently mischaracterized?

Overview

The INSTA-HLS effort includes a module that translates HLS-generated Verilog into natural language to aid debugging, but this translation has not been systematically validated. This project builds an evaluation benchmark, develops a scoring rubric, and quantifies where LLMs succeed and fail at annotating HLS output.

Semiconductor relevance

The Verilog constructs evaluated here -- registered pipelines, digital signal processing (DSP) block inference, block random-access memory (BRAM) mapping, loop-carried dependencies -- are artifacts of how HLS tools map computation onto complementary metal-oxide semiconductor (CMOS) standard cells and memory macros. Identifying LLM failure modes for these constructs is a prerequisite for commercial adoption of INSTA-HLS by Siemens electronic design automation (EDA).

Weekly plan

Weeks 1-2	Select 20-30 HLS Verilog modules from MachSuite [1] and CHStone [2]. Write ground-truth natural language descriptions covering function, pipeline structure, and resource usage.
Weeks 3-4	Develop a four-dimension scoring rubric (functional accuracy, structural accuracy, completeness, readability). Measure inter-rater reliability with Cohen's kappa on a 10-module subset.
Weeks 5-6	Prompt three LLMs (e.g., GPT-4o, Claude Sonnet, one open-weight model) with a standardized prompt. Run five independent completions per model per module to assess consistency.
Weeks 7-8	Score all annotations against the rubric. Identify failure-mode construct classes. Characterize 5-10 failure cases by error type (hallucination, missed feature, incorrect resource count).
Weeks 9-10	Synthesize results: benchmark description, rubric, results table, failure mode taxonomy, and recommendations for the INSTA-HLS annotation module. Contribute to Summer Proceedings 2026.

Deliverables

Annotated benchmark dataset, scoring rubric, LLM annotation corpus, failure mode taxonomy, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Reagen, B., et al. (2014). MachSuite: Benchmarks for accelerator design and customized architectures. IEEE IISWC, 110-119.

[2] Hara, Y., et al. (2008). CHStone: A benchmark suite for practical C-based high-level synthesis. IEEE ISCAS, 1192-1195.

Project 12: Statistical fidelity of anonymized circuit netlists for machine learning training in EDA

FAST thrust

FAST (Frontier Automation for Semiconductor Technology) §C.2.a.2 -- AI-assisted EDA design flow, synthetic and anonymized training data

Research question

Does graph-based anonymization of standard cell netlists -- replacing gate-type labels with structural equivalence-class tokens while preserving connectivity -- preserve the distributional properties that determine the difficulty and transferability of artificial intelligence (AI)-assisted placement and routing tasks?

Overview

FAST thrust C.2.a.2 identifies a key bottleneck: foundries will not share proprietary netlists, so machine learning (ML) models for electronic design automation (EDA) must train on anonymized or synthetic data. Before building anonymization pipelines, it is necessary to know whether anonymized netlists remain statistically faithful enough to train transferable models. This project quantifies that fidelity gap.

Semiconductor relevance

Standard cell netlists are the direct output of complementary metal-oxide semiconductor (CMOS) logic synthesis and the primary input to placement and routing -- the two EDA steps FAST targets for AI automation. If anonymization destroys the degree distribution, logic depth, or fan-out statistics that determine placement difficulty, ML models trained on anonymized data will not transfer to real chips.

Weekly plan

Weeks 1-2	Collect open-source gate-level netlists (OpenCores, RISC-V (Reduced Instruction Set Computer V) cores via Yosys/Sky130) and 500 circuits from Intel FloorSet [1]. Convert all to a common graph format (nodes = gates, edges = nets).
Weeks 3-4	Implement three anonymization strategies: (A) label suppression, (B) equivalence-class replacement, (C) k-anonymity graph perturbation. Apply all three to every netlist.
Weeks 5-6	Compute graph statistics (degree distribution, logic depth, fan-out, strongly connected component (SCC) count) for original and anonymized netlists. Report mean Kolmogorov-Smirnov (KS) distance per strategy as the primary fidelity metric.
Weeks 7-8	Train a graph neural network (GNN) placement policy [2] on original netlists; retrain on each anonymized variant. Measure wirelength prediction accuracy gap as a downstream task-relevant fidelity measure.
Weeks 9-10	Synthesize results into a recommendation: which anonymization strategy is safe for ML training in the FAST EDA context, and what residual fidelity gap remains.

Deliverables

Anonymized netlist dataset (three variants), fidelity scripts (Python, MIT-licensed), KS distance tables, GNN transfer accuracy results, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Mallappa, U., et al. (2024). FloorSet: A VLSI floorplanning dataset with design constraints of real-world SoC designs. arXiv:2405.05480.

[2] Mirhoseini, A., et al. (2021). A graph placement methodology for fast chip design. Nature, 594, 207-212.

Project 13: A machine learning surrogate thermal model for 3D chiplet stack co-design

FAST thrust

FAST (Frontier Automation for Semiconductor Technology) §C.2.a.3 -- chiplet and system co-design with integrated thermal management

Research question

Can a machine learning (ML) surrogate trained on synthetic floorplan configurations and open-source thermal simulations predict steady-state peak temperature for a three-dimensional (3D) chiplet stack from floorplan features alone, fast enough to guide early-stage co-design decisions without full finite-element method (FEM) simulation?

Overview

FAST thrust C.2.a.3 aims to co-design electronic and thermal performance of 3D chiplet stacks, but full thermal simulation is too slow for an electronic design automation (EDA) optimization loop. This project generates synthetic training data with HotSpot [1], extracts geometric and power features from parameterized floorplans, and trains and evaluates a fast regression surrogate.

Semiconductor relevance

The context is complementary metal-oxide semiconductor (CMOS) chiplets in a face-to-face or face-to-back 3D stack, where power delivery and heat dissipation are coupled through the same back-end-of-line (BEOL) layers used for electrical interconnect. Power density values and die footprints are drawn from published 7 nm and 5 nm CMOS node data [2].

Weekly plan

Weeks 1-2	Install and validate HotSpot 6.0 [1] against published benchmarks. Define a parameterized 3D floorplan generator (2-4 chiplet layers, 4-16 tiles each) with CMOS-realistic power densities and footprints.
Weeks 3-4	Generate 2,000-5,000 synthetic floorplan configurations; run HotSpot on each. Extract peak and mean per-layer temperature as prediction targets. Store in hierarchical data format 5 (HDF5).
Weeks 5-6	Extract feature vectors (total power, power density per layer, aspect ratio, vertical alignment, via density, distance to heat sink). Train a random forest on an 80/20 split; report mean absolute error (MAE) and 95th percentile error.
Weeks 7-8	Train a small feedforward neural network on the same features. Compare accuracy and inference time against the random forest and against HotSpot to quantify speedup in an EDA loop.
Weeks 9-10	Wrap the best model in a Python application programming interface (API). Validate on held-out configurations and write the technical report.

Deliverables

Synthetic thermal dataset (HDF5), floorplan generator and HotSpot scripts, trained surrogate models, Python prediction API (MIT-licensed), Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Huang, W., et al. (2006). HotSpot: A compact thermal modeling methodology for early-stage VLSI design. IEEE TVLSI, 14(5), 501-513.

[2] Shao, Y. S., et al. (2019). Simba: Scaling deep-learning inference with multi-chip-module-based architecture. MICRO, 14-27.

Project 14: Does Bayesian optimization find energy-accuracy Pareto-optimal stochastic computing units that differ structurally from deterministic fixed-point designs?

Research question

When Bayesian optimization searches the design space of stochastic computing (SC) units for a fixed spiking neural network (SNN) inference task, does it identify complementary metal-oxide semiconductor (CMOS) implementations on the energy-accuracy Pareto frontier that are structurally distinct from deterministic fixed-point implementations -- and if so, does the advantage arise from reduced switching activity, reduced circuit depth, or a different numerical representation?

Overview

Stochastic computing represents numbers as the probability that a random bitstream bit is 1, allowing multiplication to be implemented with a single AND gate rather than a multi-bit multiplier [1]. For SNN inference, where weights are often sparse and outputs are binary spikes, SC may offer energy advantages over fixed-point arithmetic. The conditions under which SC genuinely Pareto-dominates fixed-point are not well characterized [2]. This project uses Bayesian optimization to search the SC design space and tests whether discovered designs are structurally distinct from fixed-point baselines.

Semiconductor relevance

All candidate designs are synthesized with Yosys targeting the Sky130 process design kit (PDK) and simulated with Verilator for functional accuracy evaluation. Energy is estimated using switching activity interchange format (SAIF)-based power analysis after synthesis. The SC design space includes bitstream length, correlation management scheme, and number representation (unipolar vs. bipolar), all of which have non-obvious interactions with accuracy.

Weekly plan

Weeks 1-2	Implement a fixed-point baseline SNN inference unit in Verilog for a two-layer network on N-MNIST. Synthesize with Yosys/Sky130 and evaluate accuracy with Verilator to establish the fixed-point Pareto point (energy, accuracy).
Weeks 3-4	Implement a parameterized stochastic computing SNN inference unit with three tunable parameters: bitstream length L (16-1024), correlation management scheme (independent streams vs. shared linear feedback shift register (LFSR)), and number representation (unipolar vs. bipolar) [1,2].
Weeks 5-6	Define the Bayesian optimization loop using scikit-optimize or BoTorch: inputs are (L, correlation scheme, representation), outputs are (synthesis energy estimate, inference accuracy). Run 40-60 evaluations to map the Pareto frontier.
Weeks 7-8	Decompose energy for each Pareto-optimal SC configuration into switching activity, static power, and leakage using SAIF-based analysis. Determine whether SC wins through reduced switching activity, shallower critical paths, or reduced gate count.
Weeks 9-10	Test SC accuracy under input noise vs. fixed-point under the same noise. Write the technical report characterizing the conditions under which SC offers genuine advantages.

Deliverables

Parameterized SC SNN inference unit (Verilog), Bayesian optimization loop (Python), Pareto frontier plots, structural decomposition analysis, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Alaghi, A., & Hayes, J. P. (2013). Survey of stochastic computing. ACM TECS, 12(2s), 92.

[2] Sim, H., & Lee, J. (2017). A new stochastic computing paradigm based on non-overlap bitstream generation. IEEE Transactions on Computers, 66(7), 1134-1147.

[3] Eshraghian, J. K., et al. (2023). Training spiking neural networks using lessons from deep learning. Proceedings of the IEEE, 111(9), 1016-1054.

Project 15: Does a physics-informed neural network achieve better sample efficiency than a Gaussian-process regressor for H-NNO functional state classification from sparse experimental data?

Grant

DOE Genesis Mission -- Topic 9B -- AI-accelerated H-$NdNiO_3$ co-design

Research question

For hydrogen-doped neodymium nickelate (H-NNO) functional state classification -- distinguishing neuron, synapse, and memory-capacitor behavior from proton-configuration parameters -- does a physics-informed neural network (PINN) incorporating published density functional theory (DFT) proton-migration barriers achieve better classification accuracy than a Gaussian-process (GP) regressor when trained on fewer than 50 experimental data points, and does including the physics prior measurably reduce the number of labeled samples needed to reach 85% held-out accuracy?

Overview

The Genesis proposal (Thrust 2) proposes to build a surrogate device model for H-NNO using either physics-informed neural networks or Gaussian-process regressors, but does not compare their sample efficiency. Since experimental data from Thrust 1 arrives in batches of 20-30, the choice of surrogate architecture directly affects how quickly the closed-loop synthesis pipeline converges. This project uses publicly available H-NNO characterization data and DFT results to run a controlled sample-efficiency comparison before the hardware pipeline is running.

Semiconductor relevance

H-NNO devices are thin-film elements fabricated by sputtering on SiO_2/Si substrates with catalytic palladium (Pd) electrodes and gaps of 2-50 μm. The functional state -- correlated-insulator (neuron), metallic (synapse), or intermediate (memory capacitor) -- is controlled by proton-doping conditions (temperature, annealing time, forming-gas flow rate). This classification problem is the first step in the Genesis digital twin pipeline, and its accuracy at small sample counts determines whether the Bayesian optimization loop can outperform random synthesis-condition sampling at the month 6 go/no-go checkpoint.

Weekly plan

Weeks 1-2	Collect and curate available H-NNO experimental data: resistance-temperature sweeps, current-voltage (IV) curves, and functional state labels from published literature [1,2]. Extract feature vectors (annealing temperature, time, flow rate, channel geometry) and labels (neuron/synapse/memory). Augment with synthetic samples drawn from published proton-migration barrier ranges [3] to pre-train the PINN prior.
Weeks 3-4	Implement two surrogate models: (A) a Gaussian-process classifier using a squared-exponential kernel with automatic relevance determination (ARD) via GPyTorch; (B) a physics-informed neural network that adds a regularization term penalizing deviations from the published proton-migration energy surface [3]. Both are implemented in PyTorch.
Weeks 5-6	Run a sample-efficiency experiment: train both models on N = 10, 20, 30, 40, 50 randomly sampled data points, evaluate on a fixed held-out test set, and record classification accuracy and uncertainty calibration (expected calibration error (ECE)) for each N. Repeat 20 times with different random splits to obtain confidence intervals.
Weeks 7-8	Identify the N at which each model first reaches 85% held-out accuracy (the Genesis go/no-go threshold). Compute the ratio of N values as the sample-efficiency gain from the physics prior. Test whether the PINN uncertainty estimates are better calibrated than the GP uncertainty estimates, since calibration quality determines whether the Bayesian optimization loop will explore or exploit effectively.
Weeks 9-10	Deliver a surrogate architecture recommendation for the Genesis Thrust 2 team, with quantified sample counts and confidence intervals. Write the technical report.

Deliverables

Curated H-NNO dataset, PINN and GP surrogate implementations (PyTorch/GPyTorch, MIT-licensed), sample-efficiency plots, calibration analysis, surrogate architecture recommendation document, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Zhang, H., et al. (2022). Reconfigurable perovskite nickelate electronics for adaptive machine learning. Science, 375(6580), 533-539.

[2] Deng, S., et al. (2023). Selective area doping for Mott neuromorphic electronics. Science Advances, 9(11), eade4838.

[3] Li, B., et al. (2023). Proton migration in hydrogen-doped nickelate thin films: A first-principles study. Physical Review Materials, 7, 034601.

Project 16: Does greedy mode-assignment scheduling achieve near-optimal reconfiguration switch counts in H-NNO crossbar arrays, and at what size does it degrade?

Grant

DOE Genesis Mission -- Topic 9B -- AI-accelerated H-$NdNiO_3$ co-design

Research question

For a hydrogen-doped neodymium nickelate (H-NNO) crossbar array executing a sequence of tasks that require reassigning device modes (neuron, synapse, memory capacitor), does a greedy mode-assignment policy -- which minimizes the number of switches at each time step without look-ahead -- achieve switch counts within 20% of the exhaustive optimal, and at what crossbar size and task-sequence length does the approximation ratio first exceed that bound?

Overview

The Genesis proposal (Thrust 3) uses a reinforcement learning (RL) agent to learn reconfiguration schedules that minimize switching energy while maintaining task accuracy. But the RL agent is expensive to train and requires the digital twin to be running. A simpler question must be answered first: is greedy scheduling already near-optimal, in which case RL adds little value, or is there a significant gap that justifies the RL complexity? This project answers that question analytically and combinatorially before the RL agent is built.

Semiconductor relevance

Each H-NNO device switches between functional modes by changing its gate-voltage protocol, which takes sub-microsecond pulses at approximately 2 fJ per operation. At the crossbar scale (64×64 proposed in Thrust 2), the total switching energy per task transition depends on how many devices change mode. The optimal schedule minimizes total switches over a task sequence; the greedy policy minimizes switches at each step. This project characterizes the gap between them as a function of crossbar size, quantifying the upper bound on what RL can contribute.

Weekly plan

Weeks 1-2	Formulate the mode-assignment scheduling problem formally: a crossbar of N devices, each in one of three modes, must execute a sequence of T tasks, each requiring a specified mode distribution. The cost is the total number of mode switches. Implement an exhaustive solver (integer programming or brute-force for small N) and a greedy solver that minimizes switches at each step.
Weeks 3-4	Generate random task sequences: sample mode-distribution requirements uniformly, vary N (4, 8, 16, 32, 64 devices) and T (5, 10, 20, 50 tasks). For each (N, T) combination, run 200 random task sequences through both solvers. Record the approximation ratio (greedy switches / optimal switches) and its distribution.
Weeks 5-6	Identify the (N, T) region where greedy is within 20% of optimal vs. where it degrades. Test whether the degradation correlates with task-sequence entropy (how different consecutive tasks are) rather than with N or T independently.
Weeks 7-8	Implement two improved heuristics: (A) a look-ahead greedy that considers the next k tasks before assigning modes; (B) a Hungarian-algorithm-based assignment that treats mode reassignment as a bipartite matching problem. Measure whether either recovers near-optimal performance in the degraded region.
Weeks 9-10	Deliver a scheduling policy recommendation to the Genesis Thrust 3 team: for which crossbar sizes and task sequences is greedy sufficient, and where is a more sophisticated policy (RL or look-ahead) likely to add value? Write the technical report.

Deliverables

Mode-assignment scheduling simulator (Python), approximation ratio analysis, look-ahead and Hungarian-algorithm heuristics, policy recommendation document for Thrust 3 team, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Zhang, H., et al. (2022). Reconfigurable perovskite nickelate electronics for adaptive machine learning. Science, 375(6580), 533-539.

[2] Deng, S., et al. (2023). Selective area doping for Mott neuromorphic electronics. Science Advances, 9(11), eade4838.

[3] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2), 83-97.

Project 17: How does catastrophic forgetting severity in a simulated H-NNO reservoir network scale with reservoir size and reconfiguration frequency?

Grant

DOE Genesis Mission -- Topic 9B -- AI-accelerated H-$NdNiO_3$ co-design

Research question

In a simulated hydrogen-doped neodymium nickelate (H-NNO) reservoir network performing two-task continual learning, how does catastrophic forgetting severity -- measured as accuracy retention on task 1 after training on task 2 -- scale with reservoir size and reconfiguration frequency, and is there a reservoir size and reconfiguration rate regime where periodic mode reassignment reduces forgetting relative to a fixed-weight baseline?

Overview

The Genesis proposal (Thrust 3) targets a 30% reduction in catastrophic forgetting relative to a fixed-weight baseline, but no baseline measurement exists for H-NNO reservoir networks on continual learning tasks. This project establishes that baseline in simulation using the device surrogate from Thrust 2, systematically varying reservoir size and reconfiguration frequency to map the conditions under which reconfiguration is beneficial. The result directly informs the RL reconfiguration policy that the Genesis team will develop.

Semiconductor relevance

The simulation uses the H-NNO device surrogate from the Genesis digital twin: device parameters (resistance state, synaptic weight modulation range, switching energy) are drawn from the empirical distributions characterized in Thrust 1. The reservoir architecture follows published designs for physical reservoir computing on mem-element hardware [3], adapted to include the three-mode reconfigurability unique to H-NNO. Reservoir sizes from 64 to 512 nodes are tested, spanning the range proposed in the Genesis NAS search space.

Weekly plan

Weeks 1-2	Implement a simulated H-NNO reservoir network in Python: fixed random recurrent weights (reservoir), trainable readout layer (ridge regression), and a parameterized reconfiguration mechanism that can reassign a fraction f of devices between tasks. Parameterize by reservoir size N (64, 128, 256, 512) and reconfiguration fraction f (0, 0.1, 0.25, 0.5, 1.0).
Weeks 3-4	Define the two-task continual learning benchmark: Task 1 is NARMA-10 (nonlinear autoregressive moving-average order 10) time-series prediction; Task 2 is spoken-digit classification using the FSDD (Free Spoken Digit Dataset). Train the readout on Task 1, then on Task 2, and measure accuracy retention on Task 1 as the forgetting metric [2].
Weeks 5-6	Run the full (N, f) grid: 5 values of N × 5 values of f = 25 conditions, 10 random reservoir initializations each. Record task-1 accuracy retention, task-2 accuracy, and total switching energy (f × N × energy per switch from the device model) for each condition.
Weeks 7-8	Identify the (N, f) region where reconfiguration reduces forgetting by >30% relative to f = 0 (fixed-weight baseline) without unacceptable task-2 accuracy loss. Test whether the benefit scales monotonically with f or whether there is an optimal reconfiguration fraction that balances forgetting reduction against switching energy cost.
Weeks 9-10	Deliver a parameter recommendation to the Genesis Thrust 3 RL team: which (N, f) operating points achieve the 30% forgetting reduction target, and what switching energy budget do they require? Write the technical report.

Deliverables

H-NNO reservoir simulation framework (Python), catastrophic forgetting scaling results (N × f grid), optimal reconfiguration fraction analysis, parameter recommendation document for Thrust 3, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Zhang, H., et al. (2022). Reconfigurable perovskite nickelate electronics for adaptive machine learning. Science, 375(6580), 533-539.

[2] McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation, 24, 109-165.

[3] Tanaka, G., et al. (2019). Recent advances in physical reservoir computing. Neural Networks, 115, 100-123.

Project 22: Building an LLM-assisted chip design platform for non-expert learners

Research question

Can a browser-based platform that pairs a large language model (LLM) agent with an automated synthesis back-end enable learners with no prior hardware experience to produce functionally correct, tapeout-ready digital chip designs -- and what system prompt architecture, error feedback loop, and example corpus design produce the fewest LLM interaction turns per successful design?

Overview

Krupp et al. [1] demonstrated that high-school students with no chip design background could produce tapeout-ready designs in 90 minutes using an LLM-assisted workflow built on Tiny Tapeout. This project replicates and extends that architecture as an independent platform: a purpose-built LLM agent with a structured system prompt and example design corpus, a browser-accessible register-transfer level (RTL) simulation stage, and a GitHub Actions-driven synthesis and back-end flow. Unlike the Krupp et al. system, which targets video graphics array (VGA) output, this platform supports any simple digital design (counters, finite state machines (FSMs), serial protocols, arithmetic units), broadening the range of projects a non-expert can attempt.

Semiconductor relevance

All generated designs are submitted through the Tiny Tapeout multi-project die infrastructure using the SkyWater Sky130 130 nm open process design kit (PDK) [2]. The synthesis flow uses Yosys and LibreLane, the same open-source electronic design automation (EDA) tools used in production Tiny Tapeout submissions. Learners receive a Graphic Design Stream II (GDSII) layout file and, if the submission window aligns, a fabricated chip. This grounds the platform in real complementary metal-oxide semiconductor (CMOS) fabrication rather than simulation only.

Weekly plan

Weeks 1-2	Study the Krupp et al. architecture [1] and the Tiny Tapeout Verilog template [3]. Define the platform scope: which design categories will be supported (counter, FSM, shift register, arithmetic unit). Draft the system prompt defining the LLM agent role, constraints, coding style, and explanation register for a non-expert audience.
Weeks 3-4	Build a corpus of 8-12 paired examples (natural language behavioral description + correct, synthesizable Verilog) covering the supported design categories. Each example must pass Yosys synthesis and Verilator simulation. These serve as in-context examples for the LLM agent [4].
Weeks 5-6	Implement the chat agent user interface (UI) as a single-page web application. The UI sends user prompts and conversation history to an LLM API, displays generated RTL with syntax highlighting, and integrates a Verilator-based RTL simulation stage that returns pass/fail and waveform output without requiring testbench authorship.
Weeks 7-8	Implement the back-end integration: a GitHub repository template pre-configured with the Tiny Tapeout Yosys/LibreLane GitHub Actions workflow [3]. Build an error reporting bridge that parses synthesis log output and passes structured error messages back to the LLM agent as a follow-up prompt, enabling autonomous RTL correction.
Weeks 9-10	Run a pilot with 6-10 undergraduate non-experts. Measure: (1) first-pass synthesis success rate, (2) LLM turns per successful design, (3) error categories the agent resolved autonomously vs. those requiring human intervention. Write the technical report.

Deliverables

Chat agent UI (open-source, MIT-licensed), system prompt and example corpus, Verilator simulation integration, GitHub Actions back-end template, pilot study results (turn count, success rate, error taxonomy), Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Krupp, L., Venn, M., & Wehn, N. (2026). From RTL to prompt coding: Empowering the next generation of chip designers through LLMs. arXiv:2601.13815.

[2] Google. (2025). SkyWater-PDK: Open-source process design kit for the SkyWater 130 nm technology. github.com/google/skywater-pdk

[3] Tiny Tapeout. (2025). ttsky-verilog-template. github.com/TinyTapeout/ttsky-verilog-template

[4] Brown, T. B., et al. (2020). Language models are few-shot learners. NeurIPS, 33, 1877-1901.

Section C: Hardware-software co-design projects

Projects that combine physical device models or synthesis with automated search and AI methods

Project 18: Simulation of a memristive/memcapacitive neuromorphic crossbar

Research question

How do device-level non-idealities -- variability, endurance degradation, and sneak-path currents -- propagate to network-level inference accuracy in mixed memristor/memcapacitor crossbar arrays on a complementary metal-oxide semiconductor (CMOS) back-end-of-line (BEOL) substrate, and which network layers are most sensitive?

Overview

The intern builds a physics-grounded simulation framework for a mixed memristor/memcapacitor crossbar and studies how device non-idealities affect inference accuracy. Device models are parameterized from experimentally reported data for CMOS BEOL memristive devices.

Semiconductor relevance

Memristors and memcapacitors are emerging BEOL devices fabricated in the metal interconnect layers of a standard CMOS process. Parasitic wire resistance is drawn from CMOS metal-2/3 interconnect values, and variability distributions are taken from hafnium oxide (HfOx)- and tantalum oxide (TaOx)-based resistive random-access memory (RRAM) literature [3].

Weekly plan

Weeks 1-2	Implement the threshold-type empirical additive memristance (TEAM) memristor model [1] and Biolek memcapacitor model [2] in Python. Validate against published current-voltage (I-V) curves. Package as reusable components.
Weeks 3-4	Assemble an N×N crossbar with Kirchhoff-consistent current summation and CMOS interconnect parasitic resistance. Include both memristive and memcapacitive columns for direct comparison.
Weeks 5-6	Train a small fully connected network offline in PyTorch. Map weights into the crossbar via conductance mapping. Document how quantization and device variability interact.
Weeks 7-8	Run Monte Carlo simulations over device-to-device variability from CMOS-integrated RRAM distributions [3]. Log inference accuracy vs. variability magnitude per layer.
Weeks 9-10	Optionally reconfigure the crossbar as a fixed random reservoir with trainable readout. Write the technical report and contribute to Summer Proceedings 2026.

Deliverables

Simulation framework (Python, MIT-licensed), Monte Carlo variability analysis, accuracy-vs-degradation curves, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Kvatinsky, S., et al. (2013). TEAM: Threshold adaptive memristor model. IEEE TCAS-I, 60(1), 211-221.

[2] Biolek, D., et al. (2010). SPICE model of memcapacitor. Electronics Letters, 46(7), 520-522.

[3] Prezioso, M., et al. (2015). Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature, 521, 61-64.

Project 19: Does reinforcement learning discover novel RISC-V custom instruction extensions, or does it rediscover known ones?

Research question

When a reinforcement learning (RL) agent searches the space of custom RISC-V (Reduced Instruction Set Computer V) instruction extensions for a fixed target workload and complementary metal-oxide semiconductor (CMOS) area budget, does it converge to extensions resembling known hand-designed ones -- suggesting the hand-designed solutions are near-optimal -- or does it find structurally different extensions with better performance, indicating that the human-explored design space is incomplete?

Overview

The RISC-V ISA (instruction set architecture) allows custom instruction extensions, but these are currently designed by human experts for specific domains. This project implements a small RL-based instruction search loop and compares its outputs against published hand-designed extensions on a shared benchmark.

Semiconductor relevance

This project targets the CVA6 open-source RISC-V core [3] synthesized with OpenLane 2 / Sky130 process design kit (PDK), so every candidate extension can be evaluated for real silicon area and critical path cost.

Weekly plan

Weeks 1-2	Survey published domain-specific RISC-V custom extensions (vector, crypto, ML inference) and extract a taxonomy of structural patterns: data width, operation type, operand count [1,2]. This becomes the ground truth for comparison.
Weeks 3-4	Implement a simplified instruction search environment: a fixed benchmark (e.g., matrix multiply or convolution), a parameterized instruction template, and a reward signal combining simulated cycle-count reduction and synthesized gate count from Yosys.
Weeks 5-6	Train a proximal policy optimization (PPO) [4] agent on the search environment. Run 500-1,000 episodes. Log the Pareto frontier of cycle-count reduction vs. area overhead.
Weeks 7-8	Compare RL-discovered extensions against the taxonomy from Weeks 1-2 using structural similarity metrics. Classify each as a rediscovery, a variant, or a novel structure.
Weeks 9-10	Synthesize the most promising novel extension in CVA6 / Sky130 to obtain real area and timing numbers. Write the technical report reporting the rediscovery rate.

Deliverables

Instruction search environment (Python / Yosys), trained RL agent, extension taxonomy, Pareto frontier plots, CVA6 synthesis results, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Huang, Q., et al. (2019). Autogen: Automatic discovery of efficient deep neural network architectures via novel operations. arXiv:1910.02340.

[2] Mantovani, P., et al. (2020). Agile SoC development with open ESP. IEEE/ACM ICCAD, 1-9.

[3] Zaruba, F., & Benini, L. (2019). The cost of application-class processing. IEEE TVLSI, 27(11), 2629-2640.

[4] Schulman, J., et al. (2017). Proximal policy optimization algorithms. arXiv:1707.06347.

Project 20: Does neural architecture search rediscover von Neumann separation when processing-in-memory primitives are included in the search space?

Research question

When neural architecture search (NAS) is given a search space that includes both conventional complementary metal-oxide semiconductor (CMOS) logic primitives and processing-in-memory (PIM) resistive random-access memory (RRAM) crossbar primitives, does the search converge to von Neumann-like solutions that relegate RRAM to weight storage, or does it discover topologies where memory and compute are genuinely interleaved?

Overview

NAS has demonstrated that automated search can find neural network architectures competitive with hand-designed ones [1]. Extending the search space to include PIM primitives changes the optimization landscape fundamentally, because the energy cost of data movement is no longer separable from the cost of computation [2]. This project implements a hardware-aware NAS loop with a mixed CMOS/PIM search space.

Semiconductor relevance

PIM primitives are parameterized RRAM crossbars evaluated using NeuroSim [3], calibrated to reported silicon data for 32 nm CMOS-integrated RRAM. CMOS primitives are evaluated using Accelergy [4].

Weekly plan

Weeks 1-2	Set up NeuroSim [3] and Accelergy [4] for energy estimation of RRAM and CMOS primitives. Define a unified cost model returning energy per operation for any primitive on a common scale.
Weeks 3-4	Define the mixed NAS search space as a directed acyclic graph (DAG) of operations where each node can be a CMOS convolution, fully connected layer, PIM crossbar multiply, or data-movement operation. Implement in DARTS [5].
Weeks 5-6	Run DARTS on CIFAR-10. Log architecture weights throughout search to track whether PIM nodes are selected or suppressed. Run three independent searches with different random seeds.
Weeks 7-8	Analyze converged architectures: compute the fraction of multiply-accumulate (MAC) operations assigned to PIM vs. CMOS, and identify whether the result resembles a von Neumann partition or an interleaved topology.
Weeks 9-10	Run a control experiment with a CMOS-only search space. Compare converged energy estimates and accuracy between mixed and CMOS-only searches. Write the technical report.

Deliverables

Mixed NAS search space implementation, DARTS training logs, architecture visualization tool, energy comparison results, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. ICLR.

[2] Ielmini, D., & Wong, H.-S. P. (2018). In-memory computing with resistive switching devices. Nature Electronics, 1, 333-343.

[3] Peng, X., et al. (2019). DNN+NeuroSim. IEEE IEDM, 32.5.1-32.5.4.

[4] Wu, L., et al. (2019). Accelergy. IEEE/ACM ICCAD, 1-8.

[5] Liu, H., et al. (2019). DARTS: Differentiable architecture search. ICLR.

Project 21: Does evolutionary search discover non-obvious chiplet stacking orders that simultaneously improve thermal and bandwidth objectives?

Research question

When an evolutionary algorithm jointly optimizes chiplet arrangement and die-to-die interconnect topology for a combined thermal and bandwidth objective, does it discover stacking orders and interconnect patterns that differ qualitatively from the symmetric, layer-by-layer arrangements assumed in published 3D integration guidelines -- and if so, what physical principle explains the advantage?

Overview

Three-dimensional (3D) chiplet integration offers density and bandwidth advantages over 2D designs, but joint optimization of thermal dissipation and die-to-die bandwidth is a large combinatorial problem. Evolutionary algorithms have been applied to floorplanning [3] but not to 3D stacking order and interconnect topology jointly. This project implements an evolutionary co-optimizer and tests whether it finds qualitatively different solutions from those in the published literature.

Semiconductor relevance

Thermal evaluation uses HotSpot 6.0 [4] with complementary metal-oxide semiconductor (CMOS) chiplet power maps from published 7 nm node data. Bandwidth evaluation uses an analytical model of Universal Chiplet Interconnect Express (UCIe) die-to-die links.

Weekly plan

Weeks 1-2	Define the search space: a stack of 3-4 chiplets with binary face-to-face/face-to-back bonding decisions and variable UCIe link counts per interface. Implement HotSpot and UCIe bandwidth cost models as Python functions.
Weeks 3-4	Implement a genetic algorithm with a Pareto-front fitness function jointly minimizing peak temperature and maximizing aggregate bandwidth. Represent each candidate as a chromosome encoding stacking order, bonding orientation, and link counts.
Weeks 5-6	Run the evolutionary search for 200-500 generations with a population of 50-100. Log the Pareto front at each generation. Identify the top-10 solutions and visualize their stacking orders.
Weeks 7-8	Compare top solutions against the symmetric baseline from published 3D integration guidelines [1,2]. For any solution that outperforms the baseline, identify the structural feature responsible.
Weeks 9-10	Test whether the advantage of non-obvious solutions is robust to +/- 20% power density perturbation. Write the technical report characterizing any discovered design principles.

Deliverables

Evolutionary co-optimizer (Python, MIT-licensed), Pareto-front evolution logs, stacking order visualization tool, robustness analysis, Summer Proceedings 2026 chapter, rapid-fire research talk.

References

[1] Shao, Y. S., et al. (2019). Simba. MICRO, 14-27.

[2] Stow, D., et al. (2017). Cost analysis and cost-driven IP reuse methodology for SoC design based on 2.5D/3D integration. IEEE/ACM ICCAD, 1-7.

[3] Yan, B., et al. (2024). Learning to floorplan like human experts via reinforcement learning. DATE, 1-2.

[4] Huang, W., et al. (2006). HotSpot. IEEE TVLSI, 14(5), 501-513.