Low-Power Hybrid Analog-Digital Acceleration on Edge-Class RISC-V Platforms

Cameron Durbin[1,2], Ben Feinberg[2]

[1]University of Oregon, [2]Sandia National Laboratories

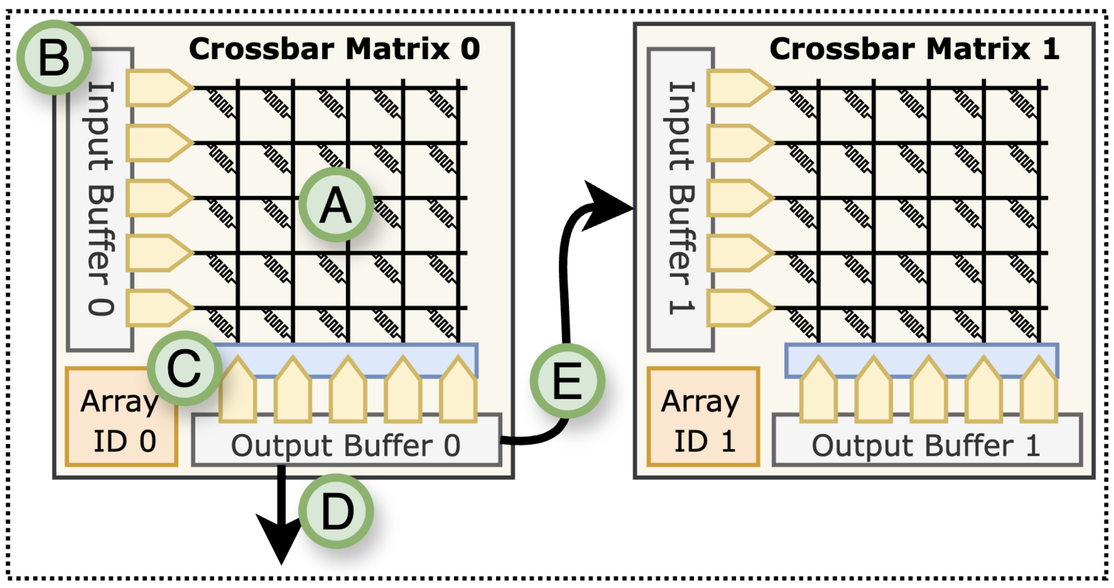
Introduction

With the end of Dennard scaling, energy-efficient acceleration is essential. Analog in-memory computing executes matrix–vector multiplications (MVMs) directly in resistive crossbars, reducing costly data movement and offering major efficiency gains. We explore a hybrid design that integrates analog arrays as coprocessors within RISC-V CPU tiles, balancing flexibility with performance.

Analog MVM Accelerators

Accelerator uses Ohm's Law and Kirchoff's Current law to perform matrix-vector multiplication. **A** is an array of resistive memory elements, **x** is an input voltage, **y** is the resulting vector summed down the bitline.

$$\text{Ohm's Law: } \mathbf{I} = \mathbf{\Sigma_j}(1/\mathbf{R_{i,j}}) \text{ x } \mathbf{V_j}$$



This work focuses on a RISC-V Edge accelerator architecture that combines general-purpose RISC-V Core and an analog coprocessor consisting of one or more analog MVM arrays in a single *Tile*.

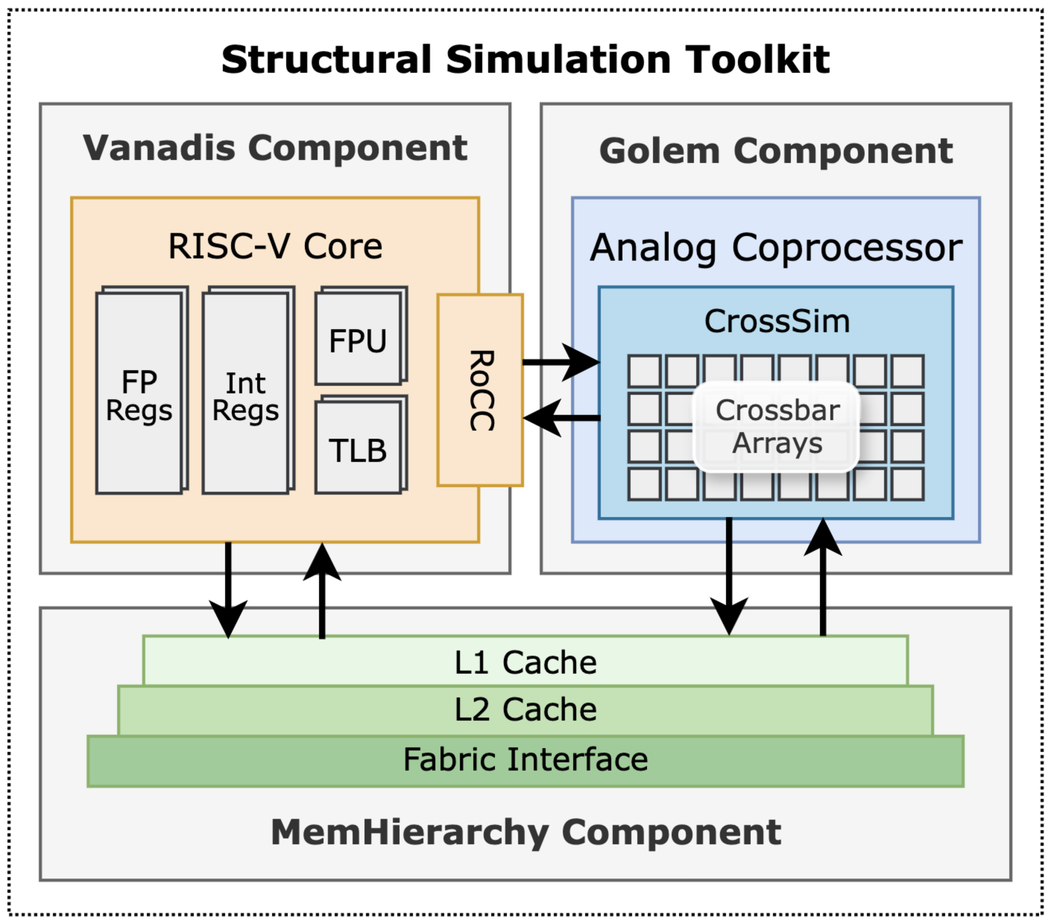
For the CPU-to-coprocessor, we adopt the conventions of the Rocket Custom Coprocessor (RoCC) interface. The RoCC interface is an extension point that lets custom coprocessors integrate directly with the CPU pipeline.

Cascading MVM operations across multiple arrays within the same tile allows for complex transformations to be executed entirely without off-chip memory accesses. Keeping data at the source of computation.

To interface with the analog array, we extend LLVM to support five new instructions and develop a custom BLAS-like library to invoke those instructions.

A	**mvm.set**	**Sets values in analog compute array**
B	**mvm.l**	**Loads the operand vector into input buffer**
C	**mvm**	**Performs analog matrix-vector computation**
D	**mvm.s**	**Stores the results values into output buffer**
E	**mvm.mv**	**Moves output buffer into a different input buffer**

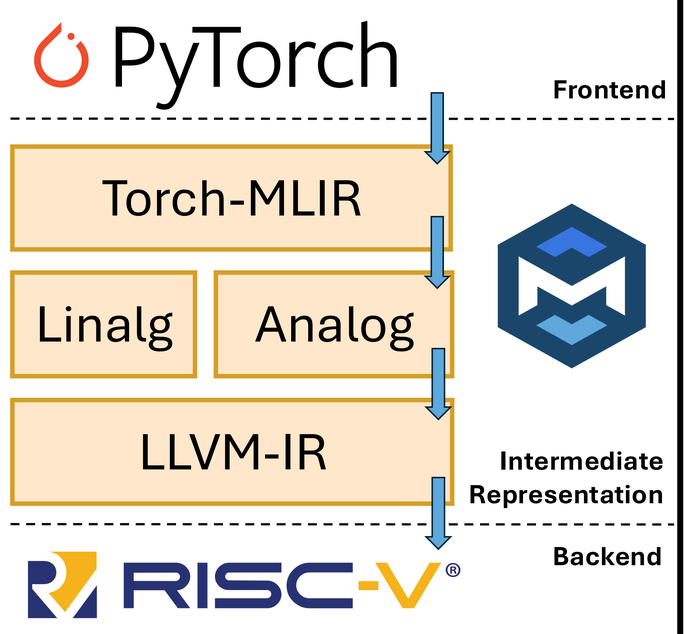
System Architecture



We use the Structural Simulation Toolkit to simulate our architecture. ***Vanadis*** models an out-of-order RISC-V core with configurable reorder buffers, pipeline widths, functional units. ***MemHierarchy*** models the private L1 and L2 cache. ***Golem*** models the analog accelerator using CrossSim.

Machine Learning

We create a custom MLIR dialect depicting our analog BLAS-like library and we swap out instances of the **Linalg** matrix-vector + 2Dconv operations with **Analog** matrix-vector + 2Dconv operations. We use Torch-MLIR to lower PyTorch models into RISC-V binaries that use our instructions.



Edge Computing

This work enables energy-efficient, low-latency, and resilient edge computing for autonomous drones and satellites.

- **Energy Efficient -** Memristor crossbars + analog accelerators reduce the energy per operation compared to digital-only CPUs/GPUs. That's critical in power-limited platforms.
- **Low Latency -** On-board MVM accelerators shorten the time from sensor input to decision without waiting for uplink/downlink.
- **Resilience -** By offloading computation to analog coprocessors, we keep digital cores free for control, fault tolerance, and communication, improving reliability.