Simulating Hybrid Analog + RISC-V Systems for HPC Applications

SuperComputing '25 International Workshop on HPC for RISC-V

November 17, 2025

Cameron Durbin[1], Jacob Flores[2], Thomas Weatherly[3], Ben Feinberg[2]

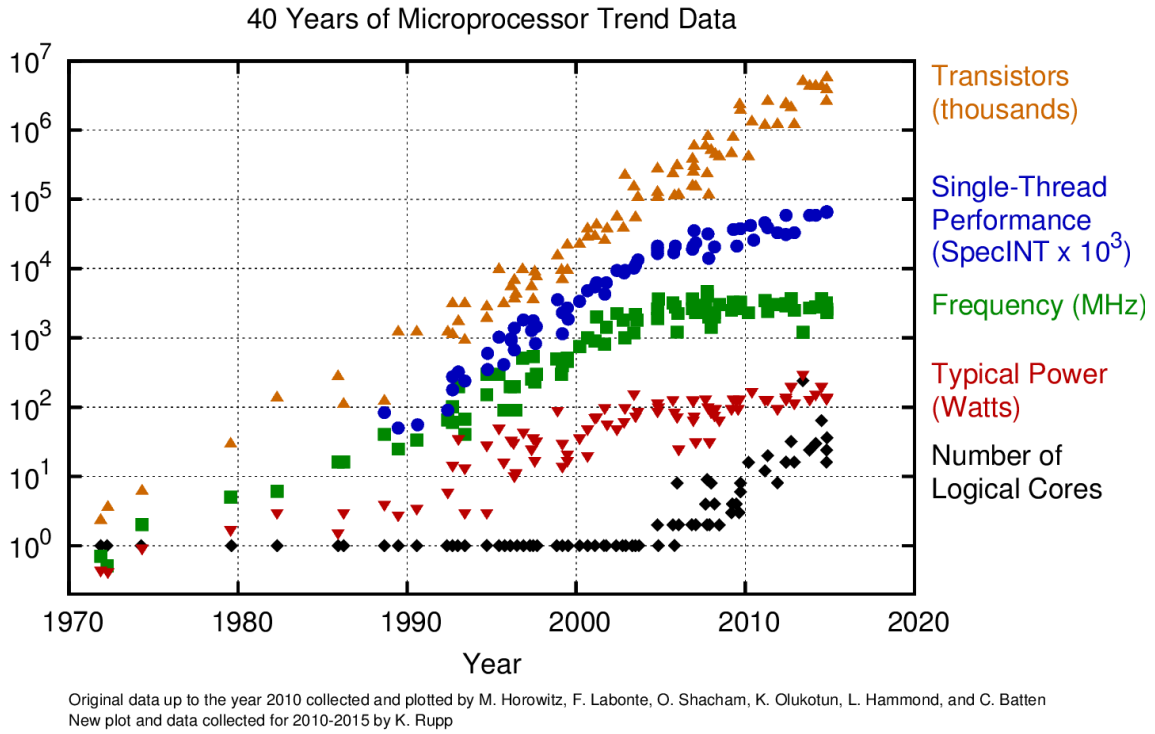
[1] University of Oregon, [2] Sandia National Laboratories, [3] Georgia Institute of Technology

Why Consider Analog?

Scaling Reality
Dennard scaling is over – more performance costs power. Most of that energy is burned moving data.

Analog Promise
Crossbar MVMs can deliver 10-100x energy/throughput gains *per MVM*.

Integration Challenge
Crossbar MVMs only pays off when it's coupled with a programmable digital system.



Core Question:
How do we make analog acceleration usable and efficient inside a full RISC-V system?

UNIVERSITY OF OREGON

Why RISC-V?

Open and Extensible ISA
Allows us to add our analog MVM instructions without proprietary toolchains.

Mature Software Stack
LLVM and OpenMP support fast, flexible HPC development on RISC-V.

Tight Integration Path
The RoCC interface enables low-overhead attachment of custom accelerators to each core.

Full-System Simulation
SST models RISC-V with cycle granularity, enabling detailed studies of analog-digital interactions.

Research Agility
RISC-V's transparent ecosystem and reproducible infrastructure make it ideal for exploring hybrid architectures.

Bringing It Together
RISC-V offers the flexibility and maturity needed to explore how analog acceleration behaves within a fully programmable system.

UNIVERSITY OF OREGON

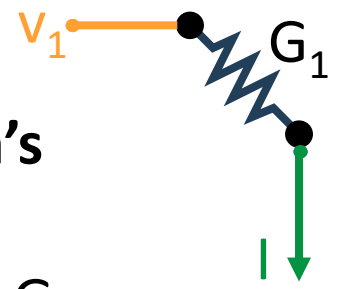
Analog MVMs

How it works

- Program the conductances of **A**
- Apply voltages of vector **x** to produce vector **y**

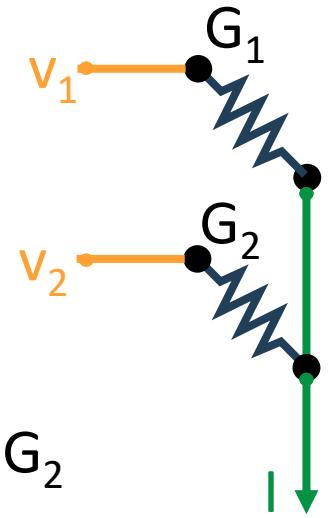
Program-once-ish

Writing resistive states is slow and consumes substantial energy.



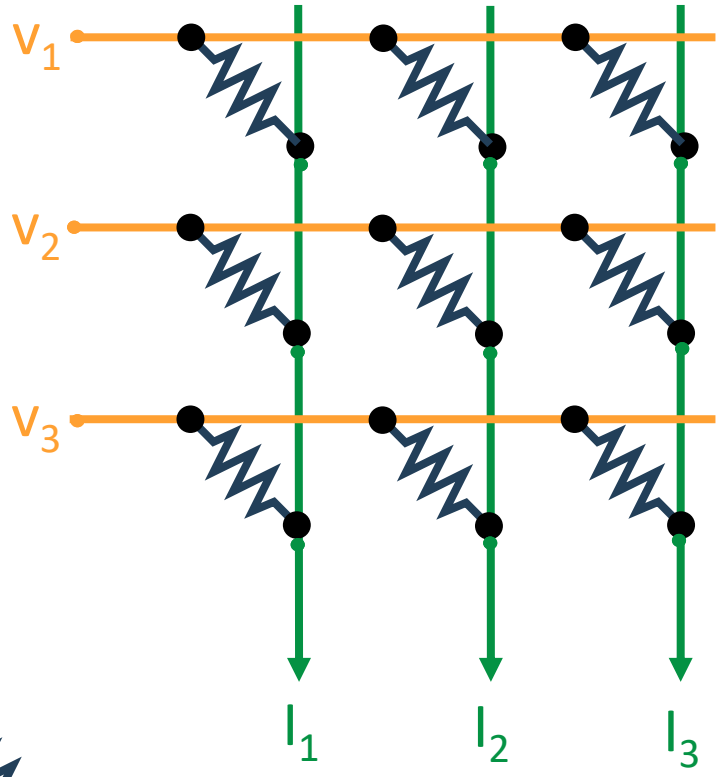
Ohm's Law

$I = v_1 G_1$



Kirchhoff's Law

$I = v_1 G_1 + v_2 G_2$



Matrix-Vector Multiplication

$\vec{I} = v\vec{G}$

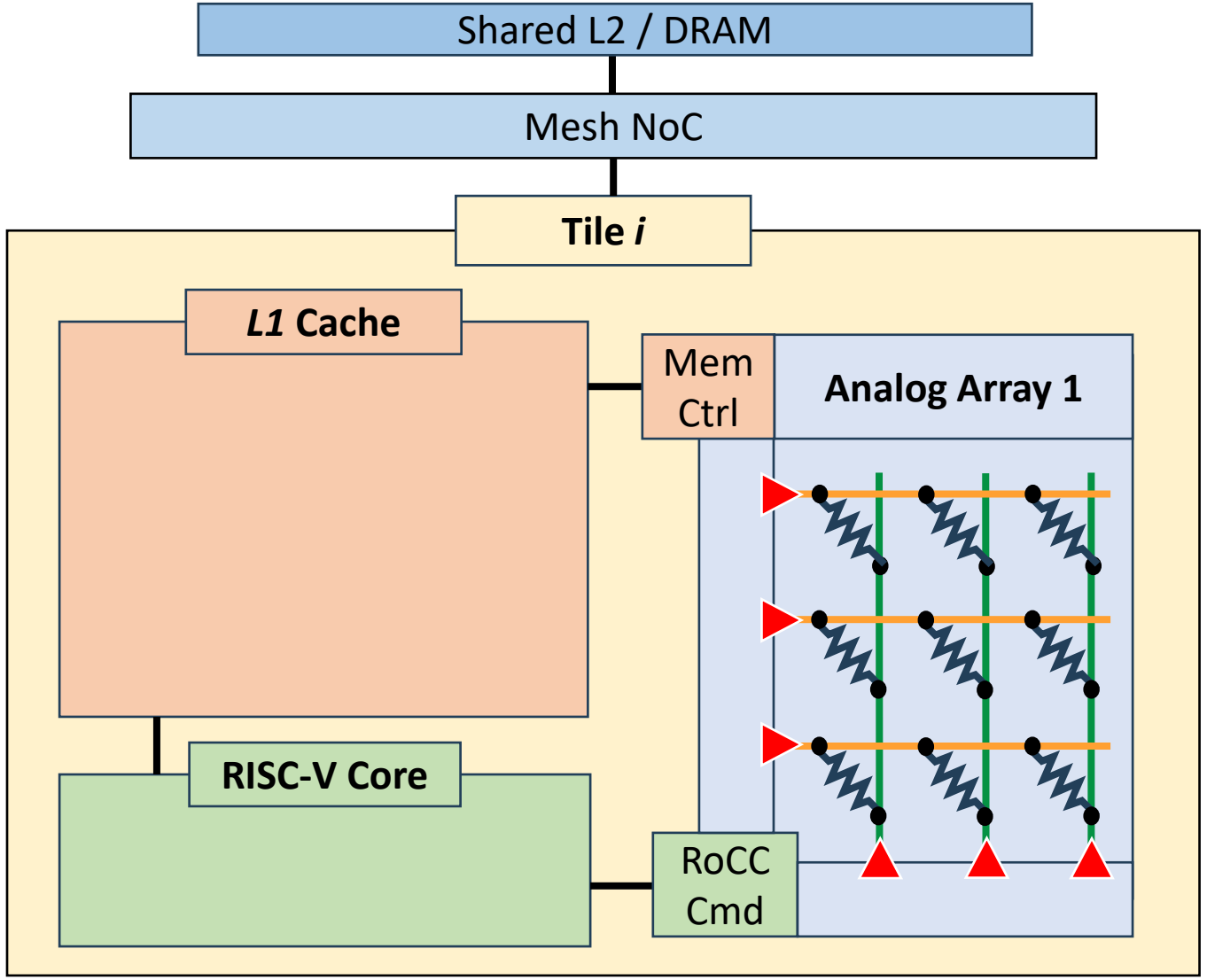
UNIVERSITY OF OREGON

System Architecture

Analog Arrays
Architectural visible input/output buffers, DMA, RoCC Interface

Tile
Contains RISC-V Core, *n* analog arrays, L1 Cache

NoC
Connects *m* tiles in a mesh, shared L2 cache, directory-based MESI coherence

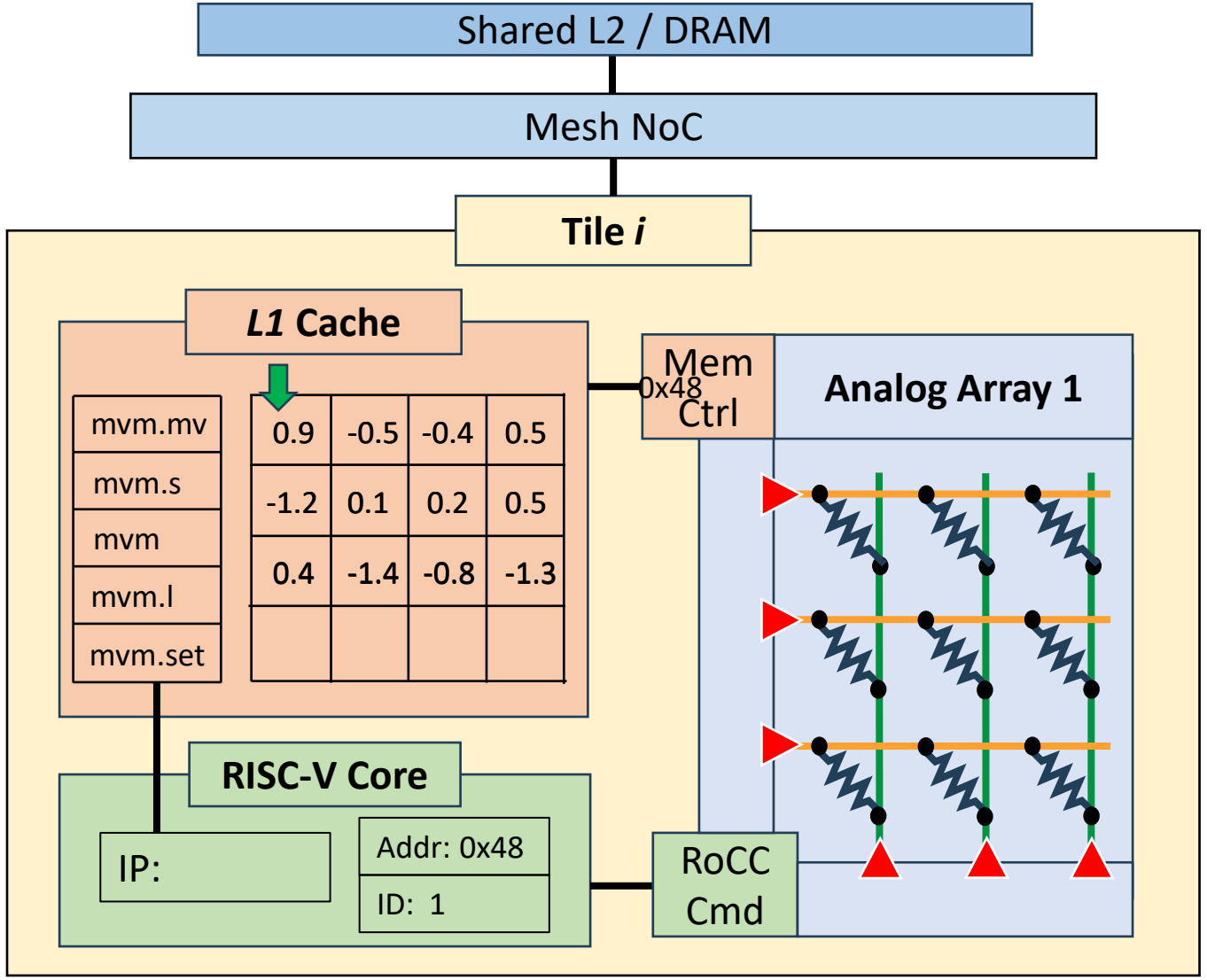


UNIVERSITY OF OREGON

Minimal Analog ISA

mvm.set %addr, id

- Program the conductances of analog array ***id*** using matrix A found at ***%addr***



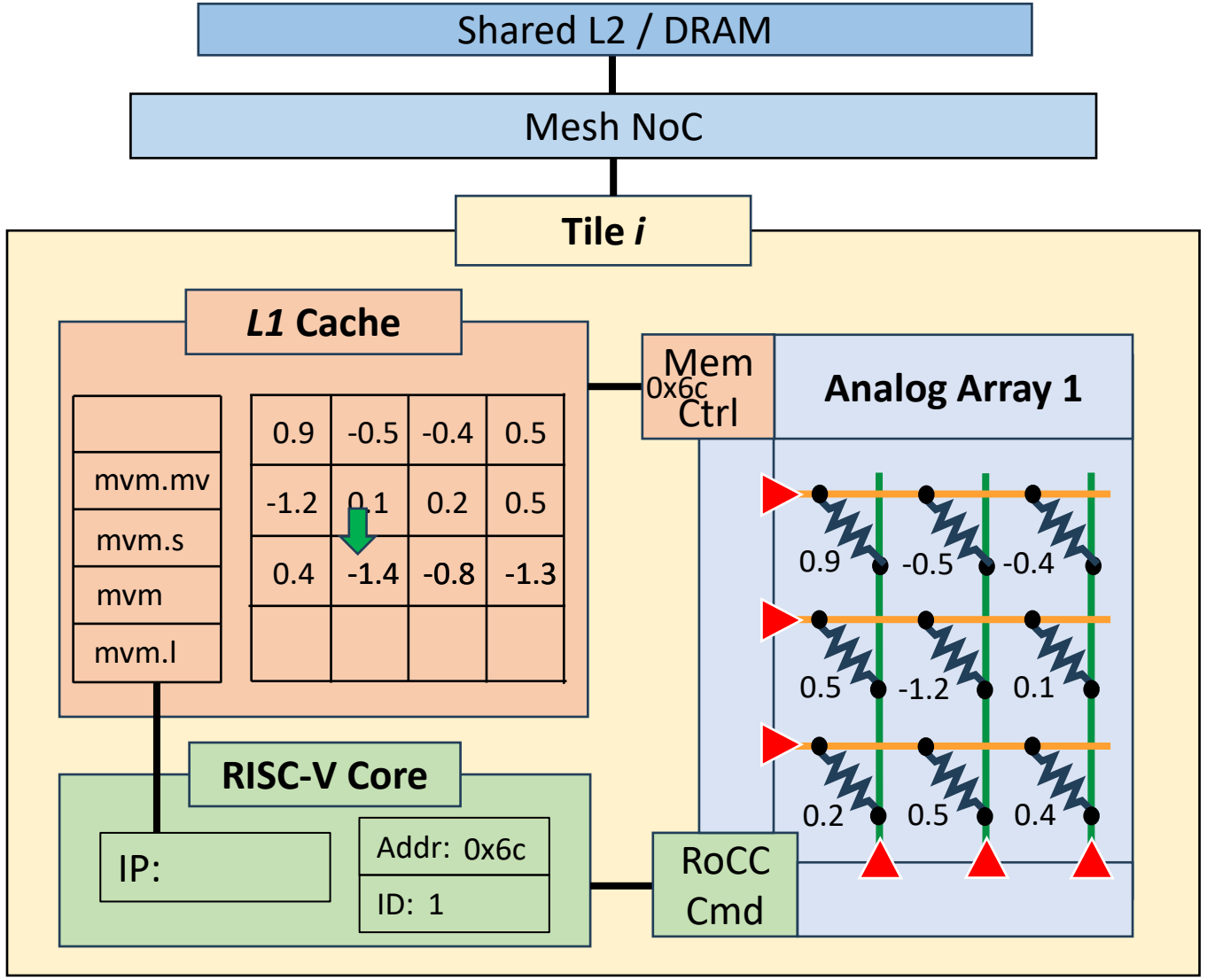
UNIVERSITY OF OREGON

Minimal Analog ISA

mvm.set %addr, id

mvm.l %addr, id

- Load an input vector into analog array ***id*** using vector x found at ***%addr***



UNIVERSITY OF OREGON

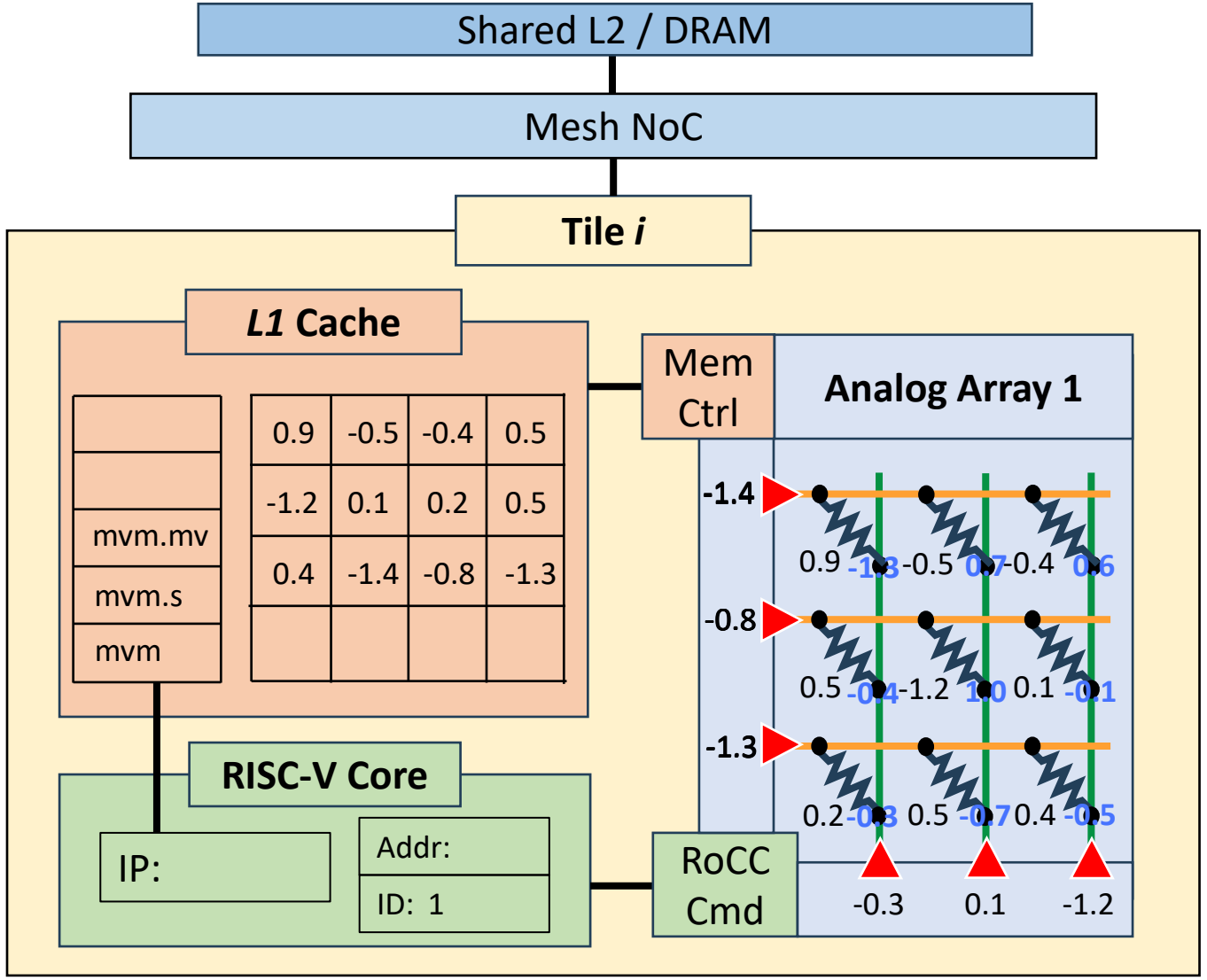
Minimal Analog ISA

mvm.set %addr, id

mvm.l %addr, id

mvm id

- Perform an MVM using the crossbar and the input buffer on array ***id***



Minimal Analog ISA

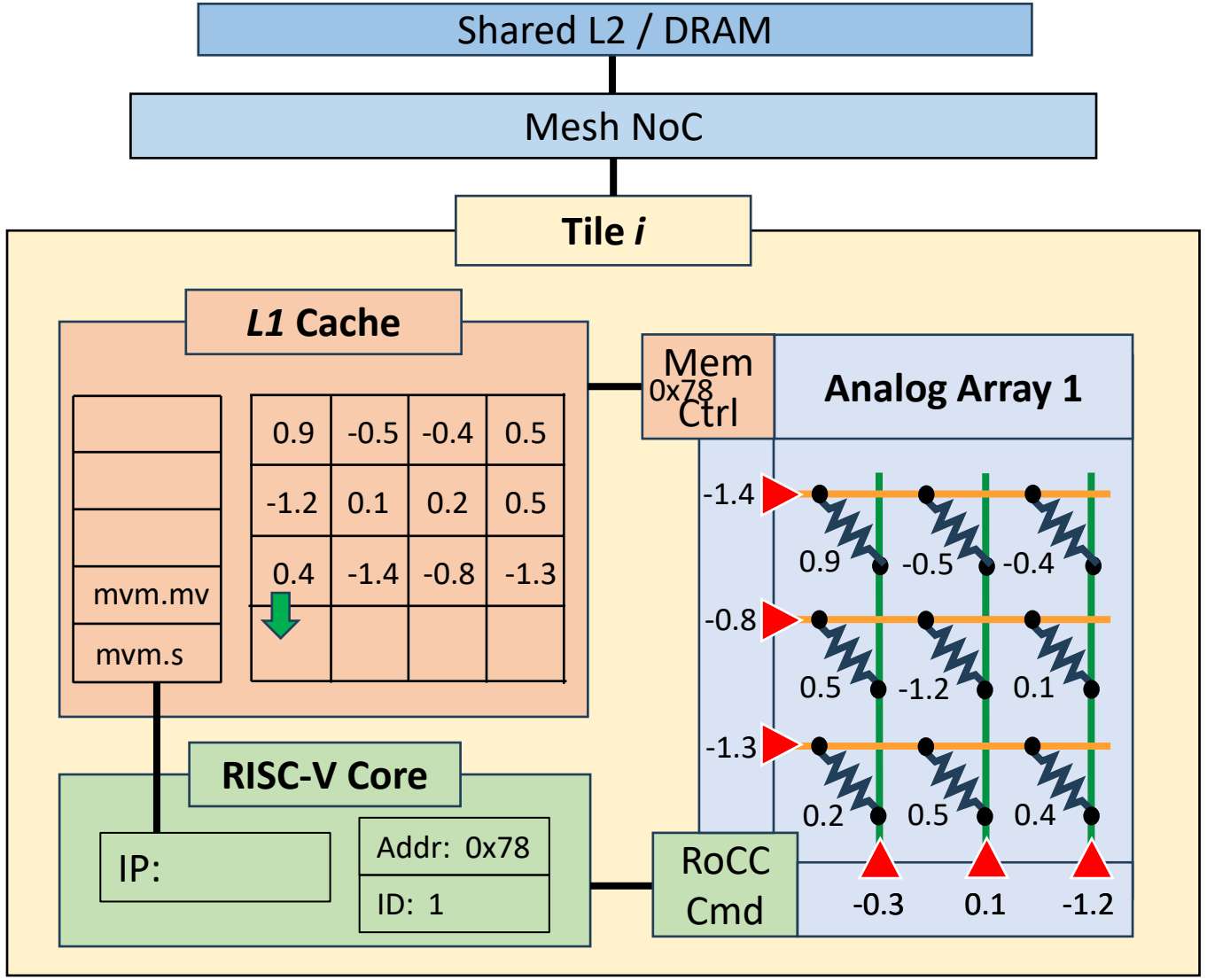
mvm.set %addr, id

mvm.l %addr, id

mvm id

mvm.s %addr, id

- Store the contents of the output buffer of array ***id*** at ***%addr***



UNIVERSITY OF OREGON

Minimal Analog ISA

mvm.set %addr, id

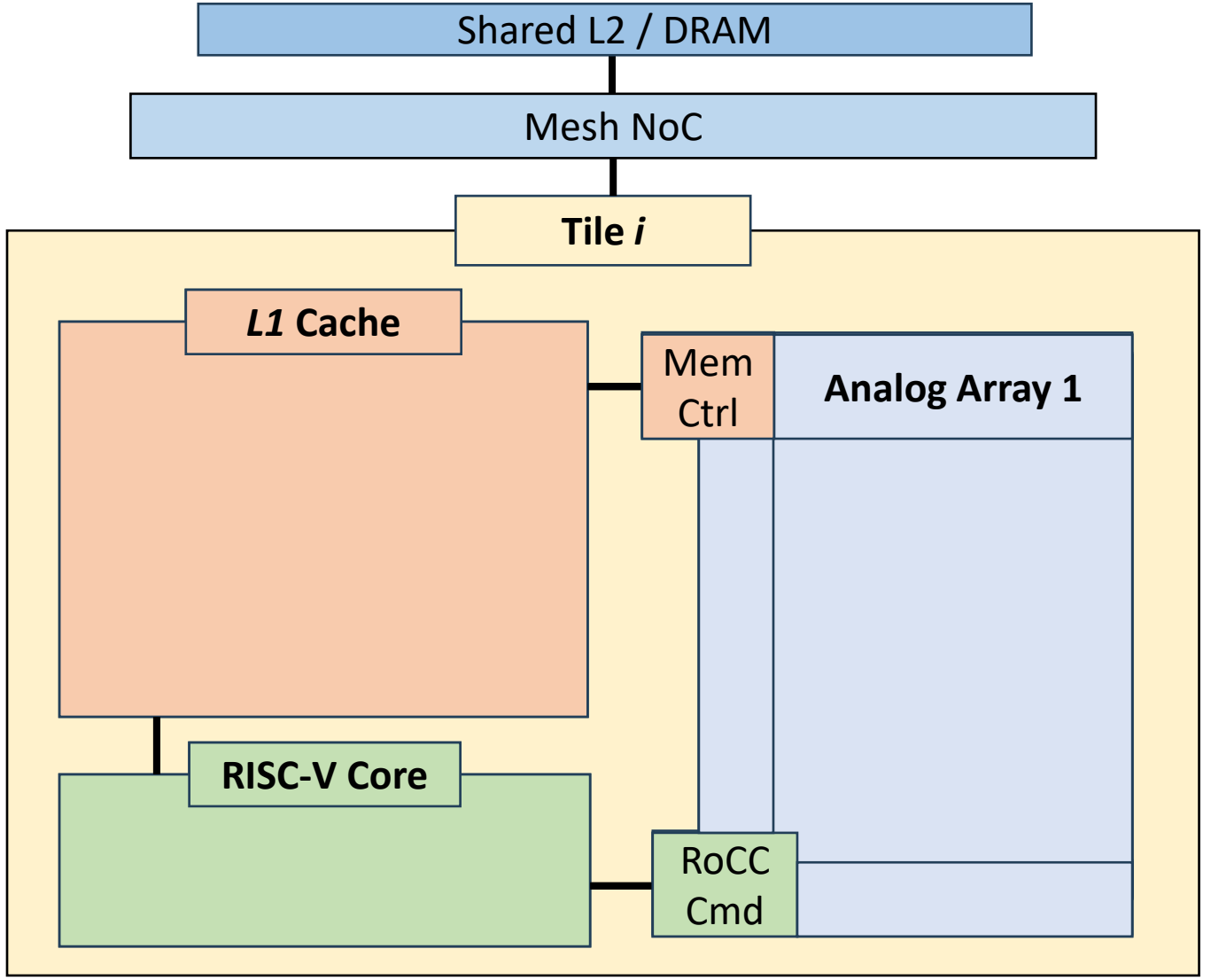
mvm.l %addr, id

mvm id

mvm.s %addr, id

mvm.mv src_id, dst_id

- Move the contents of the array ***src_id***'s output buffer to array ***dst_id***'s input buffer



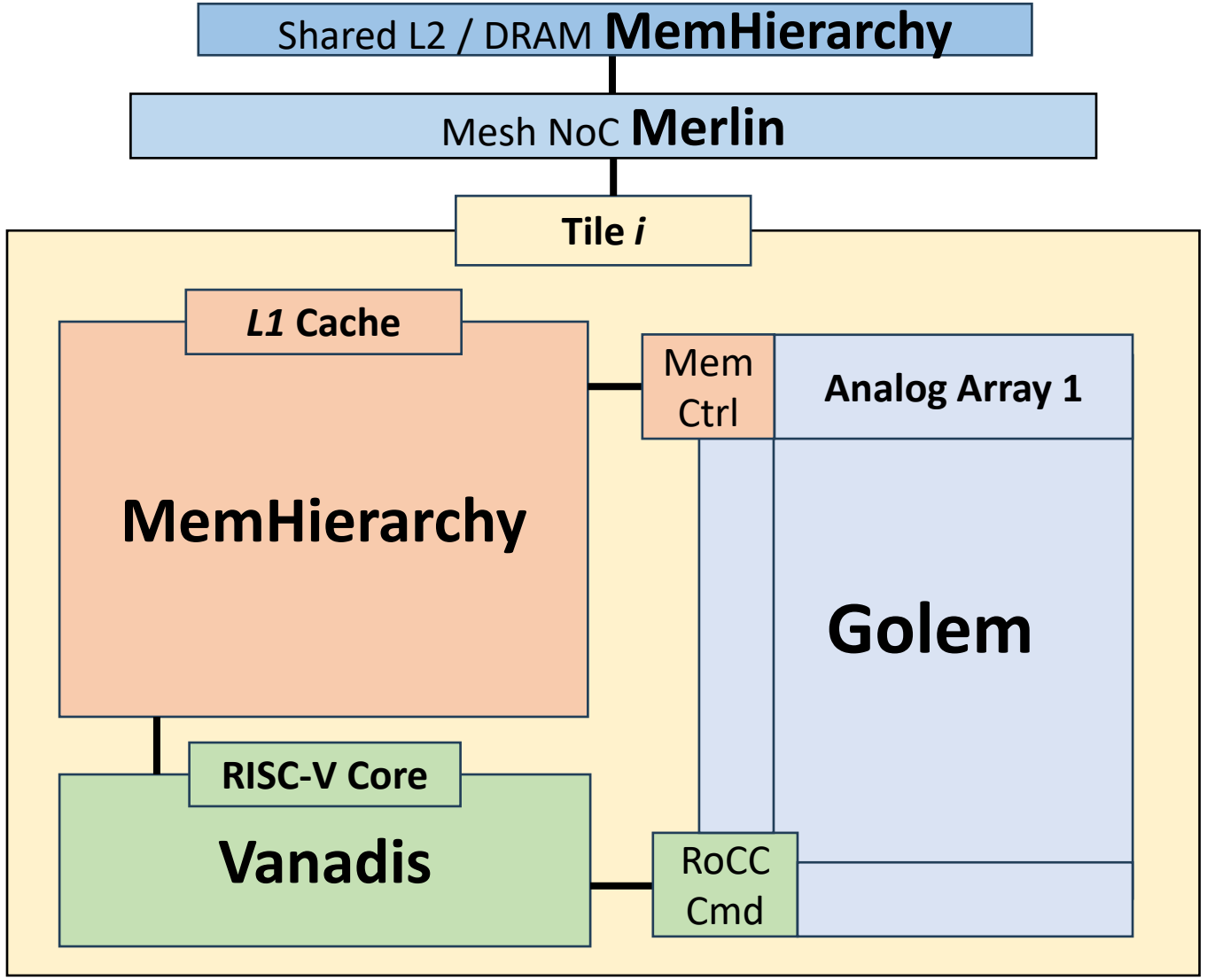
UNIVERSITY OF OREGON

Simulation Stack

Structural Simulation Toolkit (SST)

SST simulates our infrastructure using the following components:

- MemHierarchy
- Vanadis
- Golem (Optional: CrossSim)
- Merlin



UNIVERSITY OF OREGON

Software Stack

```
int mvm_set(int A, int t) {
    int s;
    asm volatile (
        "mvm.set %0, %1, %2"
        : "=r"(s)
        : "r"(A), "r"(t)
        : "memory"
    );
    return s;
}
```

```
int mvm_load(int x, int t) {
    int s;
    asm volatile (
        "mvm.l %0, %1, %2"
        : "=r"(s)
        : "r"(x), "r"(t)
        : "memory"
    );
    return s;
}
```

```
int mvm_exec(int t) {
    int s;
    asm volatile (
        "mvm %0, %1, x0"
        : "=r"(s)
        : "r"(t)
    );
    return s;
}
```

```
int mvm_store(int y, int t){
    int s;
    asm volatile (
        "mvm.s %0, %1, %2"
        : "=r"(s)
        : "r"(y), "r"(t)
        : "memory"
    );
    return s;
}
```

```
AnalogM

#pragma
for (in
    int
    int
    int
    flo

    ana
    ana
}

ma
for (in
    int
    int
    flo
    flo

    ana

    );
}
```

Standard BLAS SGEMV	AnalogBLAS SGEMV
void sgemv(	typedef struct {
char trans, →	int array_id,
	char trans,
int m, int n, →	int m, int n,
float alpha, →	float alpha,
const float *A, →	const float *A,
int lda, →	int lda,
	} AnalogMatrix;
	void analog_setmat(
	AnalogMatrix *A,
	);
	void analog_sgemv(
	AnalogMatrix *A,
const float *x, →	const float *x,
int incx, →	int incx,
float beta, →	float beta,
float *y, →	float *y,
int incy →	int incy,
);	);

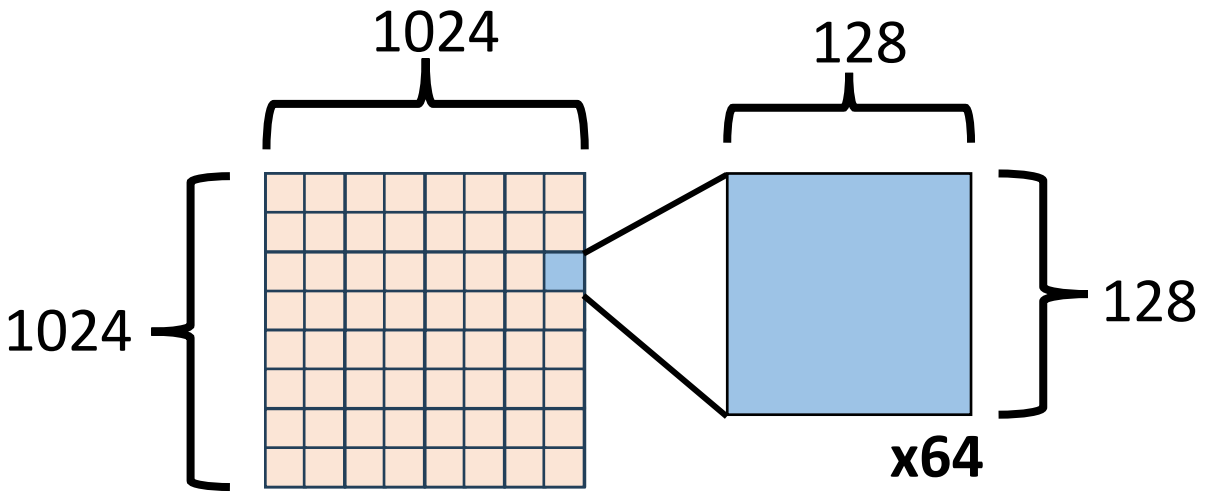
Evaluation

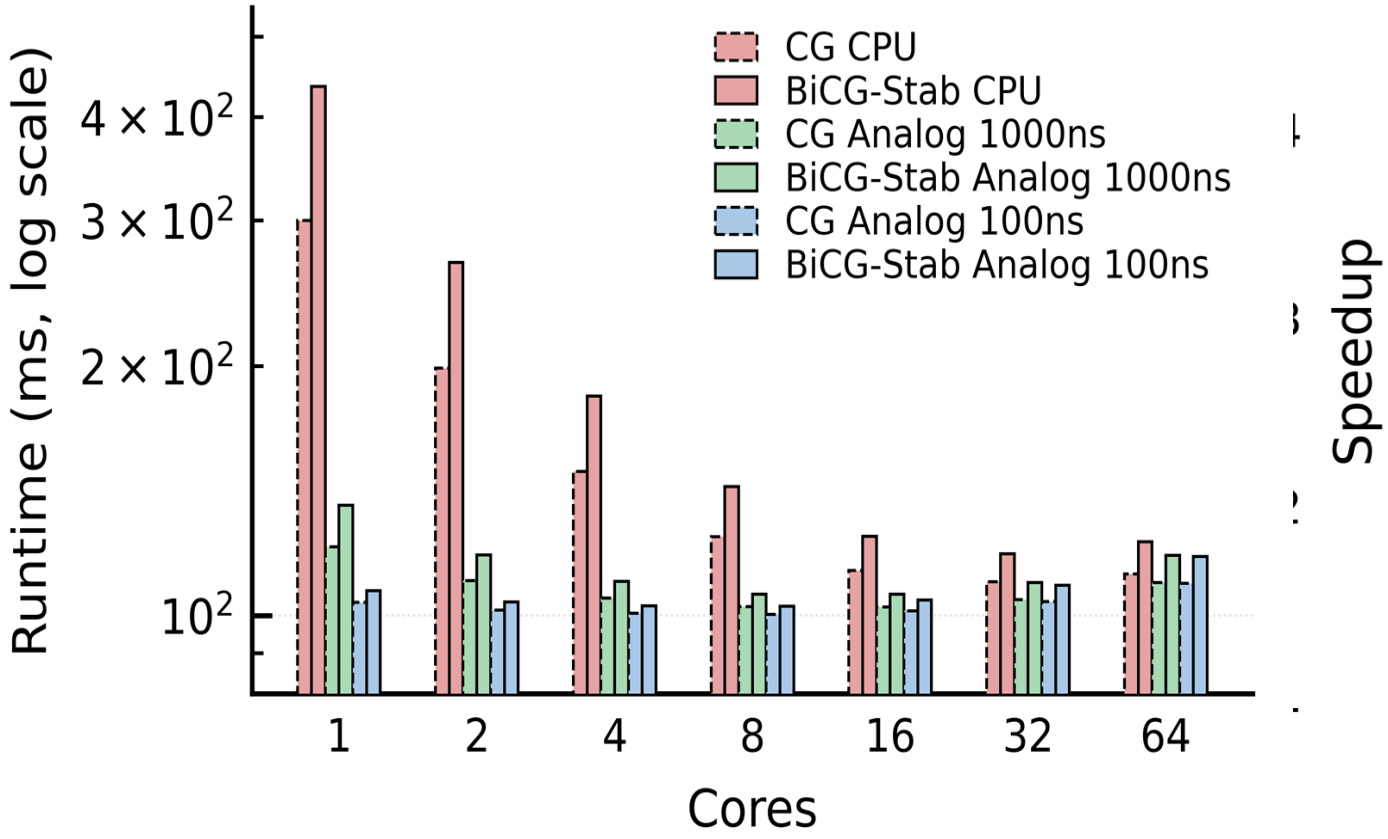
Algorithms

- Conjugate Gradient
- Stabilized Biconjugate Gradient

Setup

- **1024x1024 matrix** broken up into **64 128x128 submatrices.**
- Distribute 64 submatrices evenly across **1-64** cores.
- Use **100ns** and **1000ns** latency for MVM operations.
- Comparing CPU runtime performance vs CPU + analog MVM runtime performance.





Conclusion + Future Work

Contributions

- **First** full system simulation of a tightly coupled **analog MVM + RISC-V architecture** for HPC-style workloads.
- A **minimal, composable ISA** enabling on-tile analog pipelines.
- **Reusable golem SST component** to explore analog/digital trade-offs and non-idealities.

Future Work

- Create an MLIR dialect to automatically partition workloads across multiple arrays. Lower PyTorch models to analog compatible binaries.
- Implement co-processor scratchpads and workflows.
- Experiment with precision of analog crossbar MVMs.
- Implement the RISC-V tile on FPGA hardware and couple it with an analog coprocessor for in-silicon validation.
- Measure energy usage and performance.

UNIVERSITY OF OREGON

Questions?