RISC-V SUMMIT EUROPE • BOLOGNA • 8-12 JUNE 2026

Matrix Extensions for RISC-V Delivering on the Promise.

Four extensions, one unifying software abstraction

Dr. Philipp Tomsich (VRULL GmbH)

THE BET — SUMMIT EUROPE 2025, PARIS

One ISA cannot span edge to HPC

The RISC-V matrix effort was deliberately restructured into four complementary approaches: a family of extensions, not one rigid solution.
The reasoning was simple: no single matrix design serves a microcontroller and a supercomputer equally well, so the architecture admits several, each tuned to its end of the range. One year on, that bet is paying off.
All four are well-underway to converge on specification freeze, and the family is holding together as one software ecosystem rather than fragmenting into four.

Why a family of specifications

RISC-V spans microcontrollers to supercomputers.
Some vendors want minimal ISA disruption and software continuity.
Others want larger tiles, higher intensity, more implementation freedom.
No single design point satisfies all of them.

Where we are now

Two of the four are converging on specification freeze.
Not mutually exclusive: a vendor picks the subset that fits the target.
Shared goal: emerging narrow AI/ML types and traditional HPC precisions.

The question this talk answers: how do four hardware bets stay one software ecosystem?

A SCALABLE SOLUTION

Matrix computing is more than **AI/ML**

Embedded

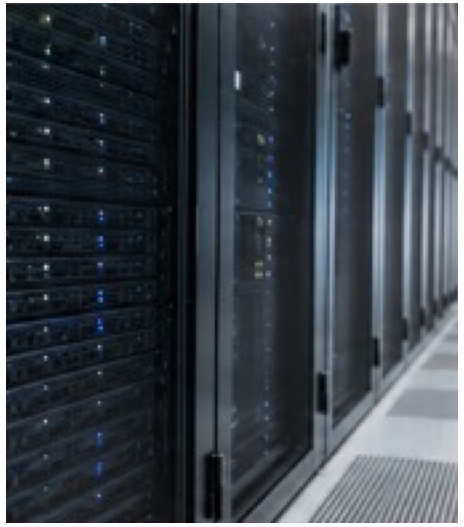
Complete GEMM on a core with just 64bit-wide vector registers. No spill. AI inference without a dedicated NPU.

- AI inference, no dedicated NPU
- Scales with VLEN (from as little as 64bit-wide vector registers)
- VL=K_EFFL one column at a time

Mid-range

Raise LMUL for K_eff depth. Partial VL balances register pressure against compute intensity.

- Modulate LMUL for deeper K
- Partial VL balances memory bandwidth against compute
- Same kernel, more throughput



Wide-VLEN HPC

LMUL = 8, full VL — bulk tiling for servers and supercomputers. Erich's end of the continuum.

- Bulk Update
- Servers to supercomputerS with the same binary
- LMUL=8, full VL: maximal tile size

A DESIGN SPACE, NOT A CONTEST

Summarizing the design space

The four specification efforts are targeting different design spaces.
Four aspects define how we should think about them — and explain why no single design point fits the whole RISC-V ecosystem.

1

RVV reuse

How much of the V register state and programming model is reused?

2

New state

Does the OS need to store new state?

3

Tile size & shape

What is the largest tile that can be supported efficiently?

This drives computational intensity and determines the performance ceiling.

4

Implementation freedom

Can we share one matrix unit across RISC-V harts?

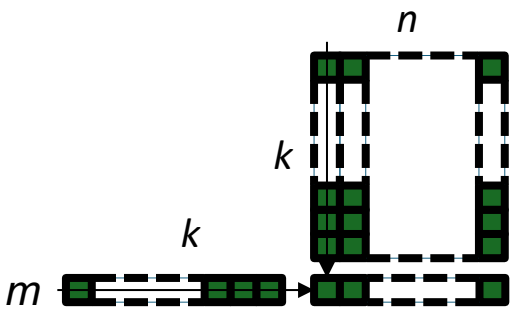
Can matrix acceleration be provided in cores that do not implement RVV?

From most RVV-integrated (BDP) to most independent (AME):
increasing capability and increasing specialization.

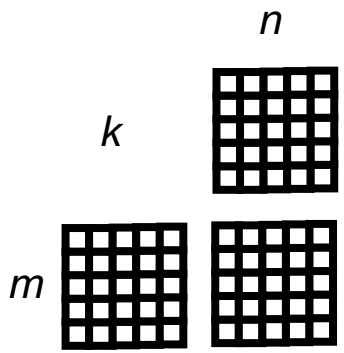
A DESIGN SPACE, NOT A CONTEST

Different approaches. Same intent.

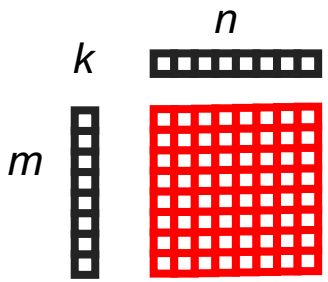
Dot-product
(fast-track)



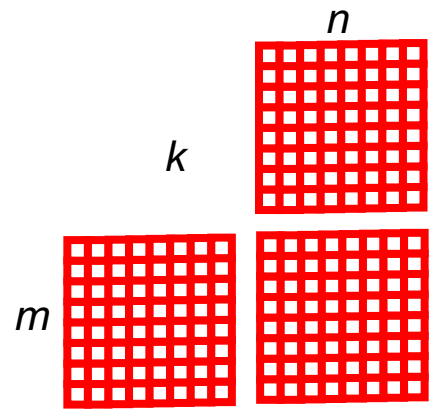
Zvvm
(Integrated Matrix)



Zvt
(Vector-Matrix)



Zvm
(Attached Matrix)



THE DEFINING QUESTION

Where do A, B, and C live?

The sharpest way to tell the four apart: are the operands and the accumulator held in RVV vector registers, or in dedicated matrix state?

	Operands A, B	Accumulator C	Abstraction
BDP	RVV vector registers	RVV vector registers	vector-matrix primitive
IME	RVV vector registers (as tiles)	RVV vector registers (as tile)	tile GEMM
VME	RVV vector registers	Dedicated accumulators	tile GEMM
AME	Dedicated tile registers	Dedicated accumulators	tile GEMM

COMPARATIVE VIEW

Match the extension to the priority

Extension	New state	Best when the priority is…
BDP	None	simplicity and minimal architectural change
IME	Minimal control (λ)	a richer matrix abstraction with strong RVV reuse
VME	Dedicated accumulators	aggressive matrix performance, with a clear vector relationship
AME	Tile + accumulator regs	a distinct matrix subsystem — shared across cores, or in cores without RVV

None of these are mutually exclusive.
A vendor can ship BDP as a lightweight add-on and a tile extension for the high end.

THE RISK

Multiple ISAs, the same math. Does software fork?

Different state, different configuration, different K limits. Left unmanaged, every AI framework forks at the ISA boundary.
This echoes the very SME-vs-AMX fragmentation issue that the RISC-V matrix-effort we set out to avoid.

SME and AMX fragmented their ecosystems with per-platform tuning and multi-version libraries.

THE RISK

Scaling the ecosystem with the hardware.

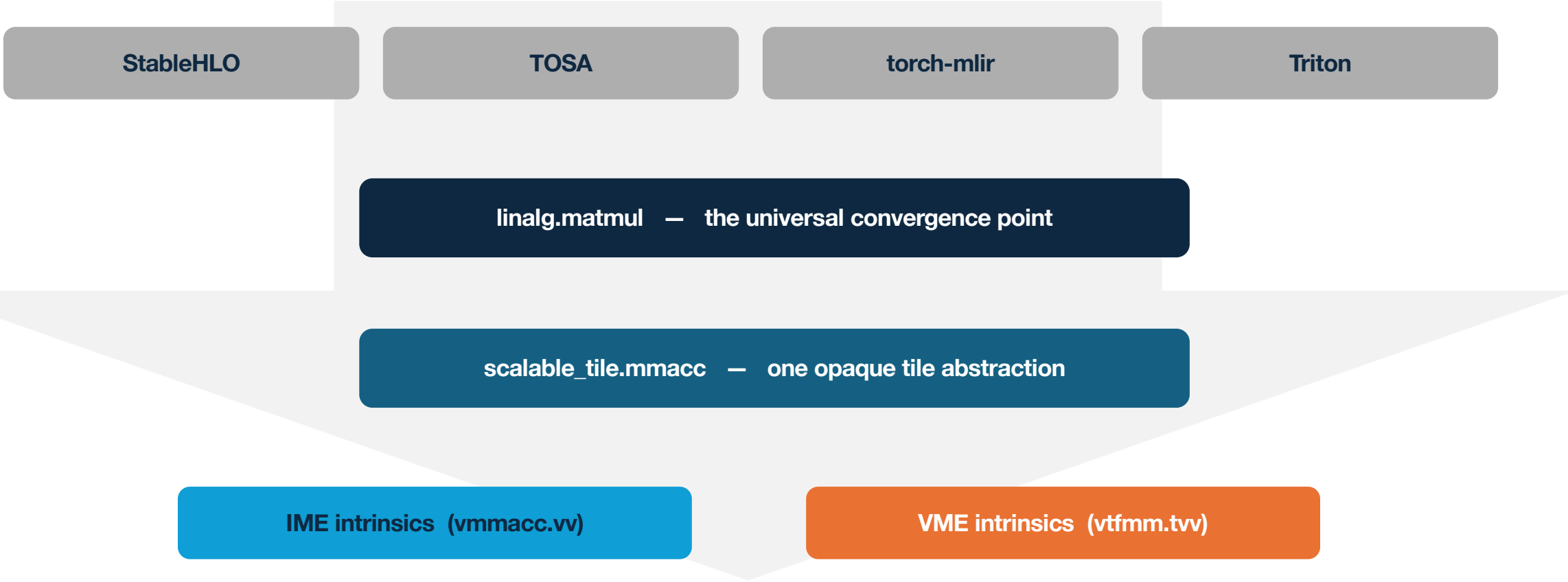
If IME and VME diverge in the toolchain, RISC-V repeats that mistake of SME and AMX.
To avoid it, the software ecosystem must scale with the hardware instead of forking.

The answer is not to pick a winner.
It is to make the hardware choice invisible to software.

THE UNIFICATION

One MLIR abstraction targets both ISAs

Every ML frontend lowers to a common abstraction: the linalg.matmul primitive.
If we lower it once to a tile op that abstracts both extensions, the cost of differentiation is abstracted away.



TWO BACKENDS VALIDATE THE ABSTRACTION

Every difference forces a design choice. The MLIR dialect must abstract these choices consistently.

Every abstraction exists because one ISA needs it and the other handles it differently.

What differs	IME (Zvvm)	VME (Zvt)	Dialect resolves by…
Tile storage	V register file	mt0..mt15 + mstatus.MS	opaque tile type
Configuration	vsetvl (one CSR)	msetmtype + msettm/k/n	config as attribute
Aspect ratio λ	1…64 — scales w/ VLEN	none (fixed TE×TE)	λ attr; gate λ≠1
Widening W	opcode, W=1,2,4,8	mtwiden, W=1,2,4	w attr; gate W=8
K depth	unbounded (λ·W·LMUL)	KMAX ≤ 4	capability query
Register alloc	standard regalloc	tile alloc + punning + spill	backend-local pass
Context save	none (V regs)	mstatus.MS, vtdiscard	backend prologue

SAME IR, TWO LOWERINGS

Resilient abstractions in the compiler

```
scalable_tile.mmacc  %c, %a, %b      // one op, type-dispatched
```

IME (Zvvm) lowering

```
vsetvl   x0, vtype(sew=16, λ=2, lmul=1)
vmtl.v   v0,  A_ptr, lda
vmtl.v   v8,  B_ptr, ldb
vmtl.v   v16, C_ptr, ldc
vfmmacc.vv v16, v0, v8
vmts.v   v16, C_ptr, ldc

→ no tile allocator, no context CSR
```

VME (Zvt) lowering

```
msetmtype mtype(tew=16,mtwiden=1),vtype
vtle16.v  mt0, A_ptr, TSS(row, ki)
vtle16.v  mt2, B_ptr, TSS(row, ni)
vtle16.v  mt4, C_ptr, TSS(row, mi)
vtfmm.tvv mt4, mt0, mt2
vtse16.v  mt4, C_ptr, TSS(row, mi)

+ tile-register alloc, + mstatus.MS mgmt
```

TAKEAWAYS

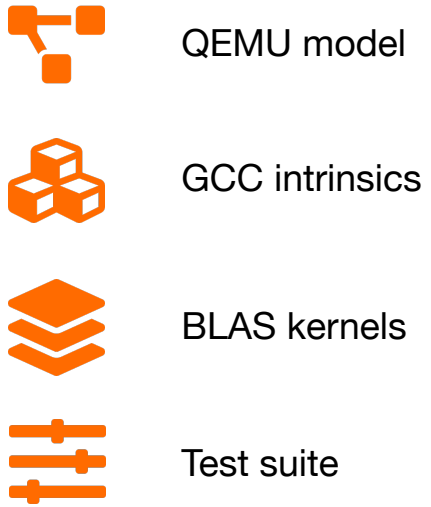
All the scaling. **One software abstraction.**

01 RISC-V matrix support is no longer a roadmap item.
The **next step is the software abstraction** that accelerates the ecosystem enablement.

02 RISC-V deliberate specified a family of four specifications for different design points: covering **from microcontrollers to supercomputer**.
All can be wrapped under **a single software abstraction**.

03 The Zvmm specification is **has concluded Internal Review**.
We expect ratification later this year.

SOFTWARE ENABLEMENT DELIVERED AS A STANDARDIZATION ARTIFACT



- QEMU model
- GCC intrinsics
- BLAS kernels
- Test suite

Co-developed with the specification.
Lowering the barrier for implementers, accelerating the upstream adoption in open-source, and making the specification testable from the start.