A RISC-V VPU for Very Long and Sparse Vectors

Gopinath Mahale, Karim Charfi, Tejas Limbasiya, Teresa Cervero, John Davis
{gopinath.mahale, karim.charfi, tejas.limbasiya, teresa.cervero, john.davis}@bsc.es
Barcelona Supercomputing Center (BSC), Barcelona, Spain

MEEP | MareNostrum Experimental Exascale Platform

https://meep-project.eu/

Motivation

- Vector Processing Unit (VPU) -> reduced instruction memory, reduced address translations and deeper pipelines -> high compute performance

Challenges:

- Very long vectors? -> Limitation of register space on Vector Register Files (VRF) -> energy constraints
- Energy-efficiency in sparse-vector computations? -> not cache-friendly

VecProM: Vector Processor, under MEEP Project

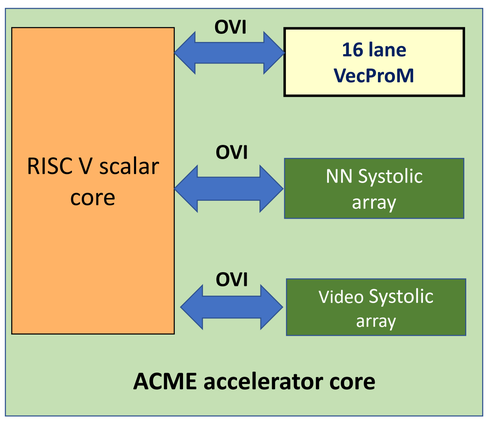


Fig 1. ACME accelerator core

- **Accelerated Compute and Memory Engine (ACME)** -> MEEP[1] project at BSC
- ACME accelerator core consists:
  - A RISC-V scalar core
  - A RISC-V VPU (VecProM)
  - Two systolic arrays
- VecProM is a modified version of a baseline RISC-V VPU [2] -> support for very long vectors and sparse vectors
- Disaggregation of memory and arithmetic instruction execution
  -> Improvement in performance and energy savings
- A specialized high-bandwidth memory path -> bypasses the cache hierarchy and connects VecProM to a scratchpad memory, termed **Long Vector Register File (LVRF)**
- LVRF -> Acts as virtual VRF for the physical VRF
  -> Maintains 32 vector registers as per the RISC-V ISA specifications.
- Loads and stores on LVRF -> **Micro-engines (MEs) ->** On-the-fly configuration

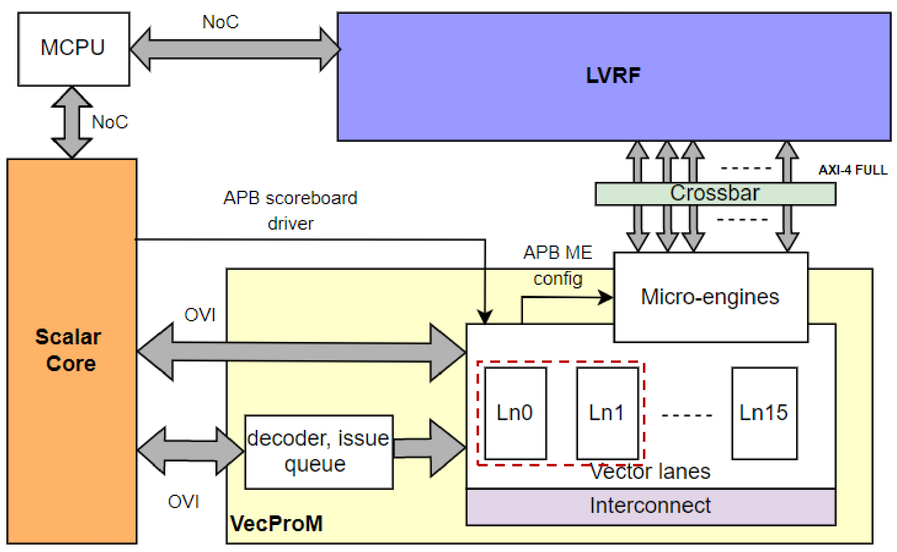


Fig. 2. VecProM interfaced with a scalar core and LVRF

VecProM operation

Two operand-dependent modes:

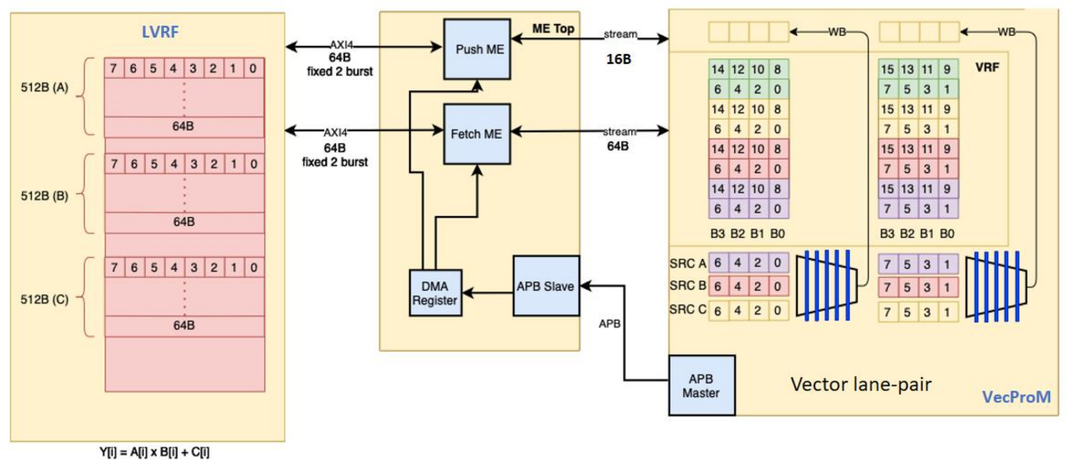
Mode_SD:

- Suitable for short dense vectors, i.e., VL fits on the VRF
- Instructions, operands and results over OVI

Mode_LDSp:

- Suitable for very long dense vectors and sparse vector operands, which do not benefit much from cache hierarchy
- Instructions over OVI, operand and results stored in LVRF, accessed through Micro-engines
- Hardware strip-mining
- MCPU [1] gathers non-zero elements of sparse data -> packed as dense data -> stored in LVRF

Micro-engines and Mode_LDSp



DMA Registers

	32 bits registers						
0x0	Reserved						start[0]
0x4	v_length[31:18] (in $ lines)				v_size[17:0] (in $ lines)		
0x8		valid src_2	valid src_1	valid src_0	dest [19:15]	src_2 [14:10]	src_1 [9:5]	src_0 [4:0]

Fig. 3. Configuration and functioning of Micro-engines

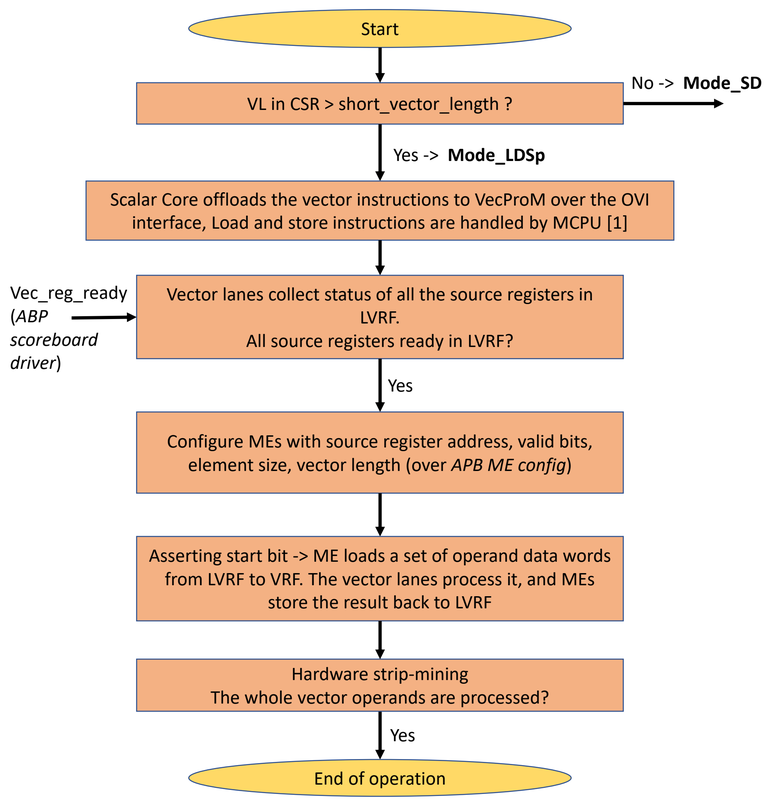


Fig. 4. Sequence of events during instruction execution

- Introduced minimal changes to the existing data path of baseline, thus retaining functionality and benefits of the baseline.
- The modes of operation mainly differ in memory path
- Provides flexibility to switch between the modes, based on the operand-type

References

[1] Fell, A., Mazure, D. J., Garcia, T. C., Perez, B., Teruel, X., Wilson, P., & Davis, J. D. (2021). The MareNostrum Experimental Exascale Platform (MEEP). *Supercomputing Frontiers and Innovations*, *8*(1), 62–81. https://doi.org/10.14529/jsfi210105
[2] https://www.bsc.es/research-and-development/projects/epi-european-processor-initative-epi

The MEEP project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 946002. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Croatia, Turkey.

MEEP Partners: Barcelona Supercomputing Center (Centro Nacional de Supercomputación); University of Zagreb Faculty of Electrical Engineering and Computing; TÜBİTAK BİLGEM