

Ventaglio: At-the-Roofline Sparse Tensor Contractions on Vector Processors for Transformer Inference

Bowen Wang*, Chi Zhang*, Diyou Shen*, Renzo Andri†, Navaneeth Kunhi Purayil*, Luca Benini*‡

**Integrated Systems Laboratory, ETH Zürich;*

†Computing Systems Lab, Huawei;

‡DEI, University of Bologna

PULP Platform

Open Source Hardware, the way it should be!



@pulp_platfor



pulp-platform.org



youtube.com/
pulp_platform

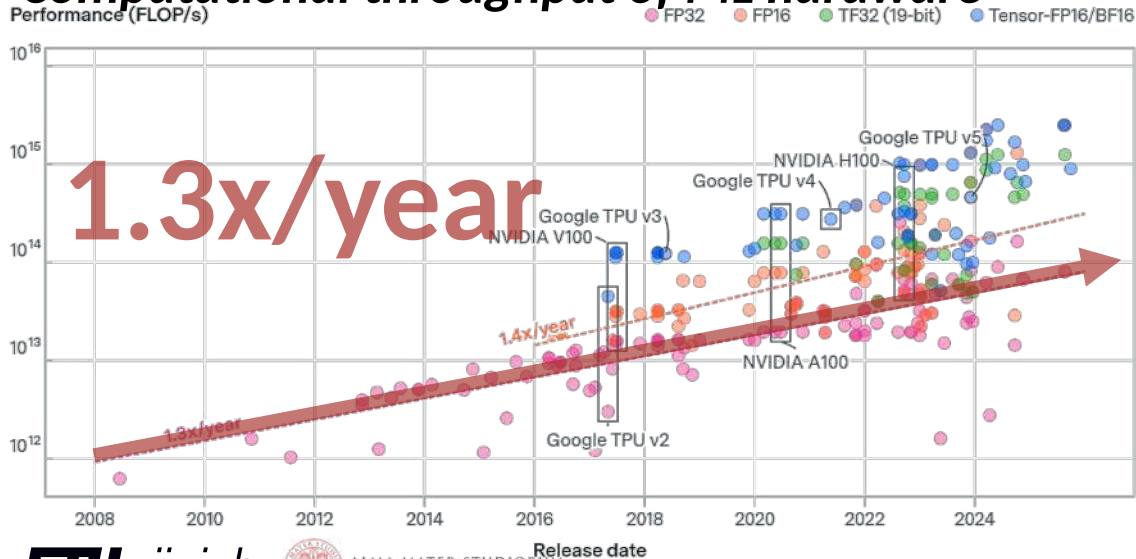


AI Hardware Platforms: Model Scaling Law

Number of parameters of frontier AI models

2.4x/year

Computational throughput of ML hardware



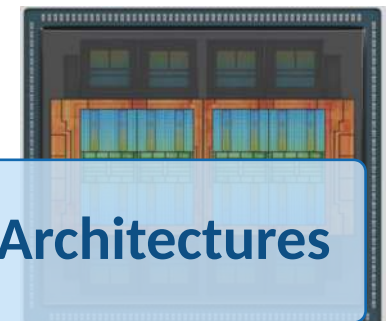
- Modern ML platforms provide **massively parallel computing capabilities** through system scaling

NVIDIA Blackwell^[1]

16,869 Scalar Cores + 528 Tensor Cores

AMD INSTINCT MI355X^[2]

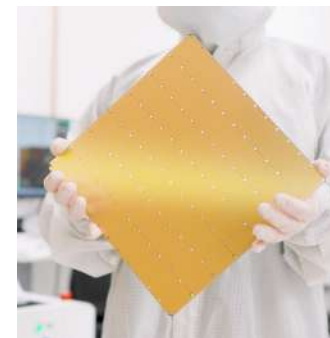
16,384 Scalar Cores + 1024 Tensor Cores



Massively Parallel Architectures

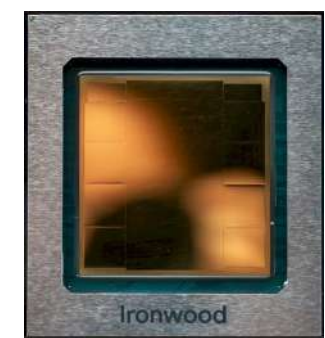
Cerebras WSE-3^[3]

900,000 Scalar Cores



Google TPU Ironwood^[4]

Scale out to 9,216 chips



[1] T. Ajay, et al. "Nvidia blackwell platform: Advancing generative ai and accelerated computing." HCS '24.

[2] S. Alan, et al. "AMD Instinct™ MI300 series modular chiplet package-HPC and AI accelerator for exa-class systems." ISSCC '24.

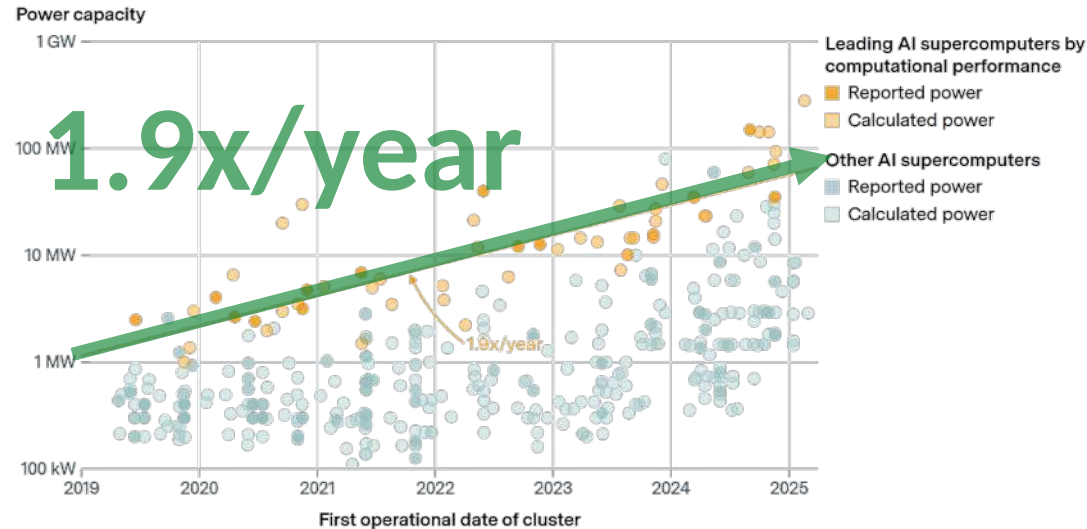
[3] I. Lie, Sean. "Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning: Cerebras systems." HCS '22.

[4] J. Norman, et al. "Ironwood: Delivering Best in Class perf, perf/TCO and perf/Watt for Reasoning Model Training and Serving." HCS '25.

AI Hardware Platforms: Rising Energy Cost



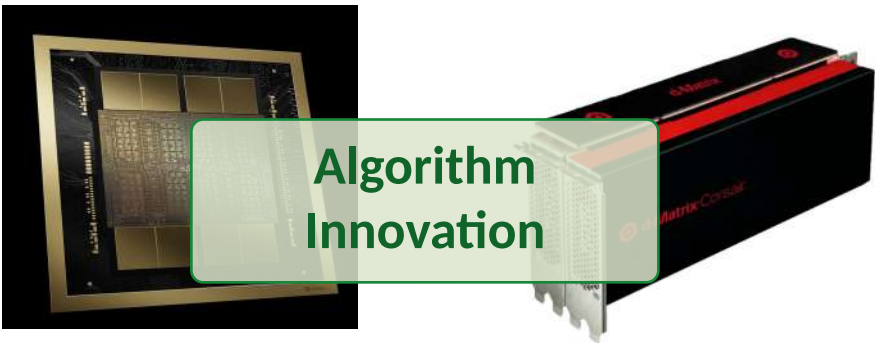
Power requirements of leading AI supercomputers



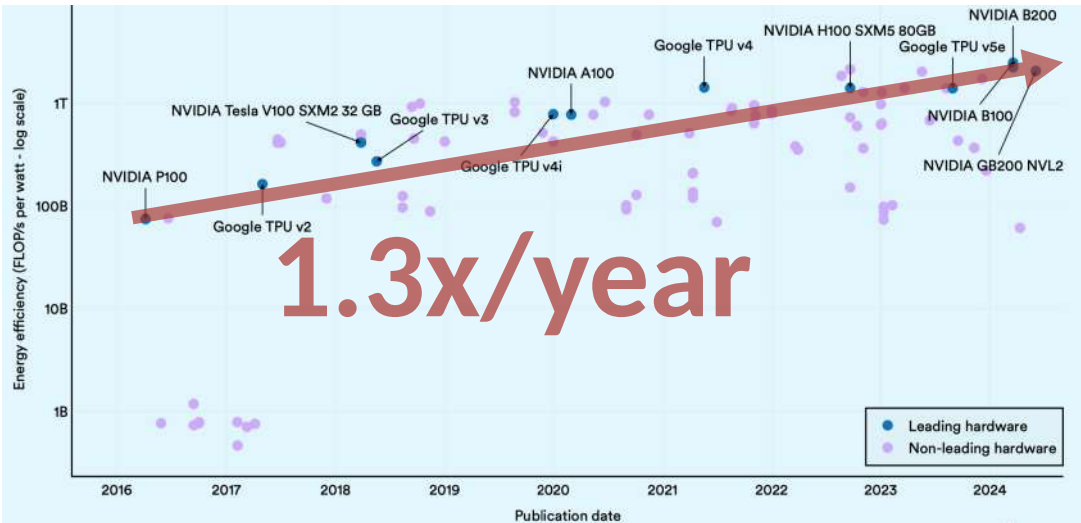
➤ Driven by reducing **data movement**, **computing specialization**, and exploiting **numerical co-design**

NVIDIA Blackwell
FP4/FP8 Tensor Cores

d-Matrix Corsair
NPU-style CIM



Energy Efficiency of leading ML hardware



Tenstorrent Blackhole^[1] Google TPU Ironwood

Dataflow-aware routing Systolic array-based dataflow

Hardware Specialization



[1] BVasiljevic, Jasmina, and Davor Capalija. "Blackhole & tt-metalium: The standalone ai computer and its programming model." HCS '24.

A Fast-Evolving Model Zoo: Model Compression Techniques



- Compression trades dense, regular computation for **sparse, irregular** data access
- **Static sparsity** (fixed offline): pruning, low-rank, vector quantization
 - SparseGPT, Wanda, AQLM, VPTQ

Sparse access granularity and target set footprint

A Fast-Evolving Model Zoo: Model Compression Techniques



- Compression trades dense, regular computation for **sparse, irregular** data access
- **Static sparsity** (fixed offline): pruning, low-rank, vector quantization
 - SparseGPT, Wanda, AQLM, VPTQ
- **Contextual sparsity** (per token): activation sparsity, dynamic depth, MoE
 - Deja Vu, PowerInfer, Mixture-of-Depths, Mixtral, DeepSeek-MoE

Sparse access granularity and target set footprint

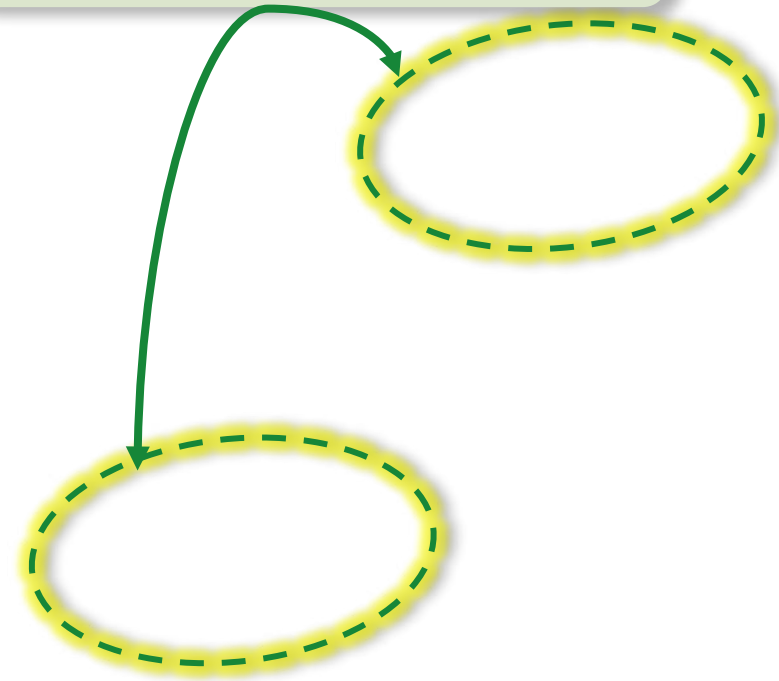
A Fast-Evolving Model Zoo: Model Compression Techniques



- Compression trades dense, regular computation for **sparse, irregular** data access
- **Static sparsity** (fixed offline): pruning, low-rank, vector quantization
 - SparseGPT, Wanda, AQLM, VPTQ
- **Contextual sparsity** (per token): activation sparsity, dynamic depth, MoE
 - Deja Vu, PowerInfer, Mixture-of-Depths, Mixtral, DeepSeek-MoE
- **Sequence sparsity** (per query): sparse attention, KV-cache compression
 - StreamingLLM, H2O, Quest, NSA, MoBA, KIVI, SnapKV

Sparse access granularity and target set footprint

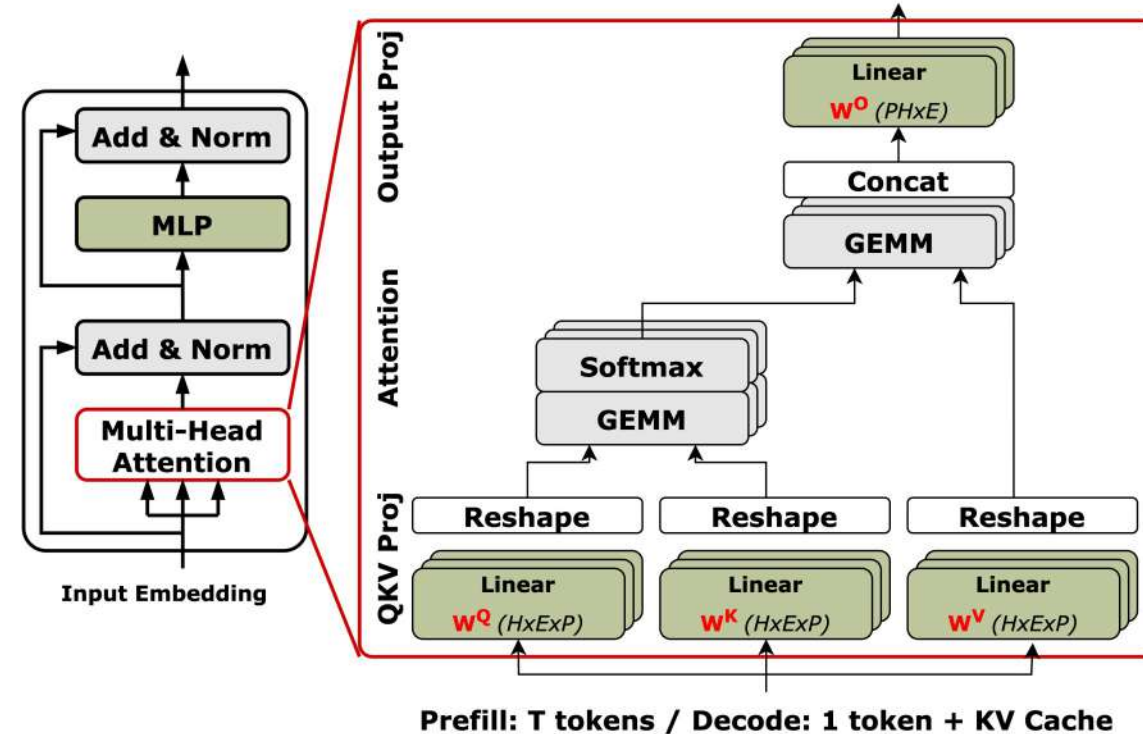
Focus of this work



Sparse Tensor Contraction: Prefill & Decode

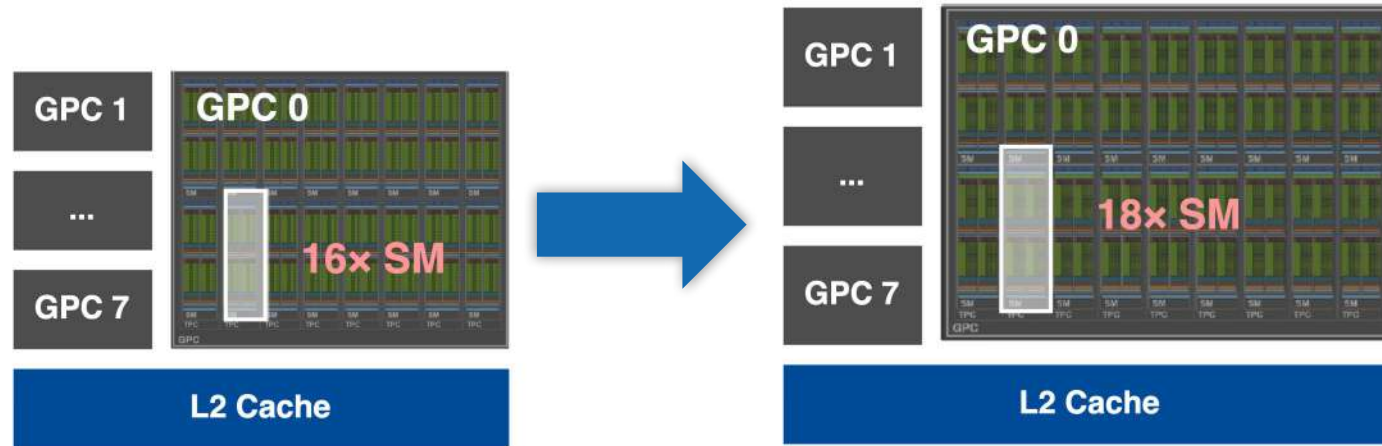


- Same computational graph, two regimes.
- **Prefill:**
 - Feed a matrix of T tokens at once
 - Linear and attention layers are GEMM-orientated
 - Weights are reused T times → **compute-bound**
- **Decode:**
 - One token per step
 - GEMM collapses to GEMV
 - Weight and KV byte is read once per token → **memory-bound**



*Efficient sparse tensor contraction
is required at both matrix-matrix and matrix-vector granularities*

GPU Sparsity Support: From NVIDIA Ampere to Rubin



Ampere (NV A100)

108 SMs per GPU

4x 3th gen tensor cores (312 TOPS*),
2:4 structured, static
FP16/BF16/INT8 (2:4), INT4 (4:8)

Blackwell (NV GB200)

144 SMs per GPU

4x 5th gen tensor cores (1248 TOPS*),
2:4 structured, static
+ NVFP4 / FP8 / FP6

Rubin (NV VR200)

224 SMs per GPU

4x 5th gen tensor cores (35/50 POPS**) + 3rd-gen Transformer Engine
2:4 compressed, **adaptive**;
+ **runtime zero-skipping**

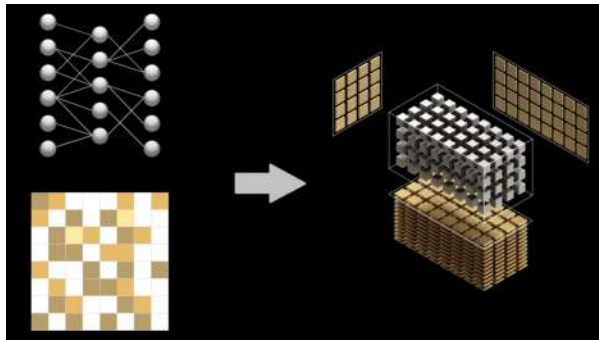
Compute Engine Considerations

Native HW Acceleration for Model Compression

- ML models are updated on a **weekly basis**
 - Model sparsification and quantization promise substantial efficiency gains
 - Rapid evolution demands that hardware can adapt **fast**

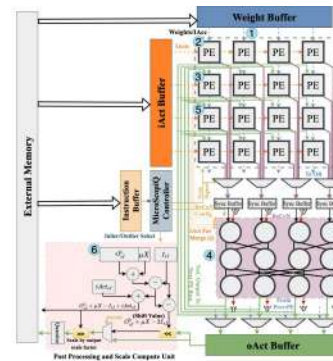
NVIDIA Sparse Tensor Core

Support mapping 2:4 structured sparsity format



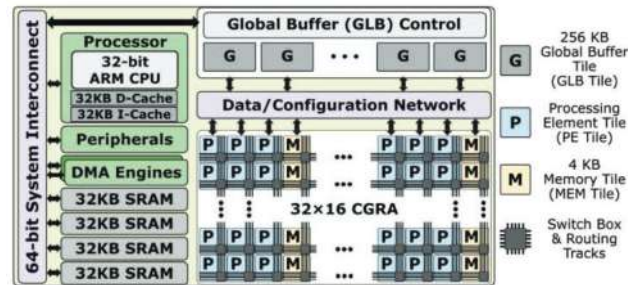
MicroScopiQ^[1]

WS systolic array for outlier-aware quantization



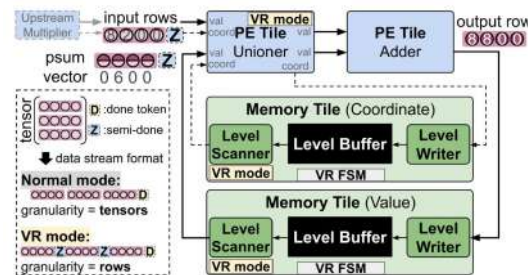
Onyx^[2]

Programmable accelerator for sparse tensor algebra



Opal^[3]

CGRA for Full Sparse Machine Learning



Research trends:

Bulky Accelerators

Format rigidity, Low reusability,
Underutilization in inference

FPGA, CGRA-based Designs

Offer programmability but at the cost of
energy efficiency

We need **configurable specialization**
to bridge HW flexibility and efficiency

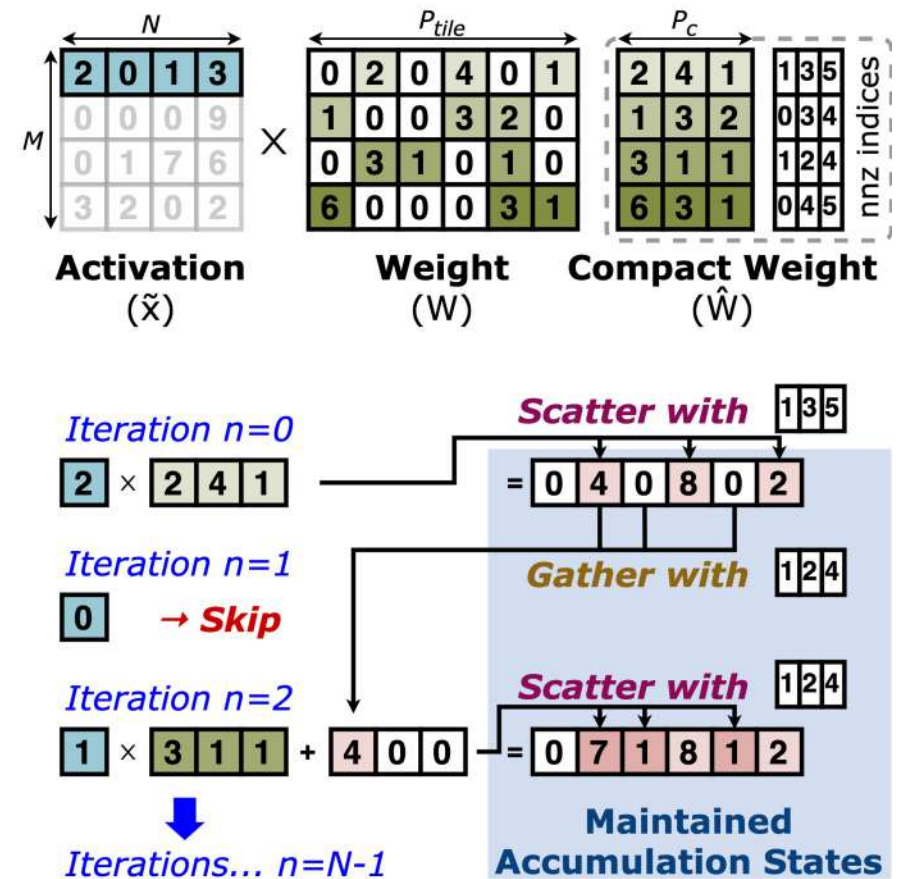
[1] R. Akshat, et al. "MicroscopiQ: Accelerating foundational models through outlier-aware microscaling quantization." ISCA '25.

[2] K. Kalhan, et al. "Onyx: A 12nm 756 GOPS/W coarse-grained reconfigurable array for accelerating dense and sparse applications." VLSI '24.

[3] K. Jahyun, et al. "OPAL: Outlier-Preserved Microscaling Quantization Accelerator for Generative Large Language Models." DAC '24.

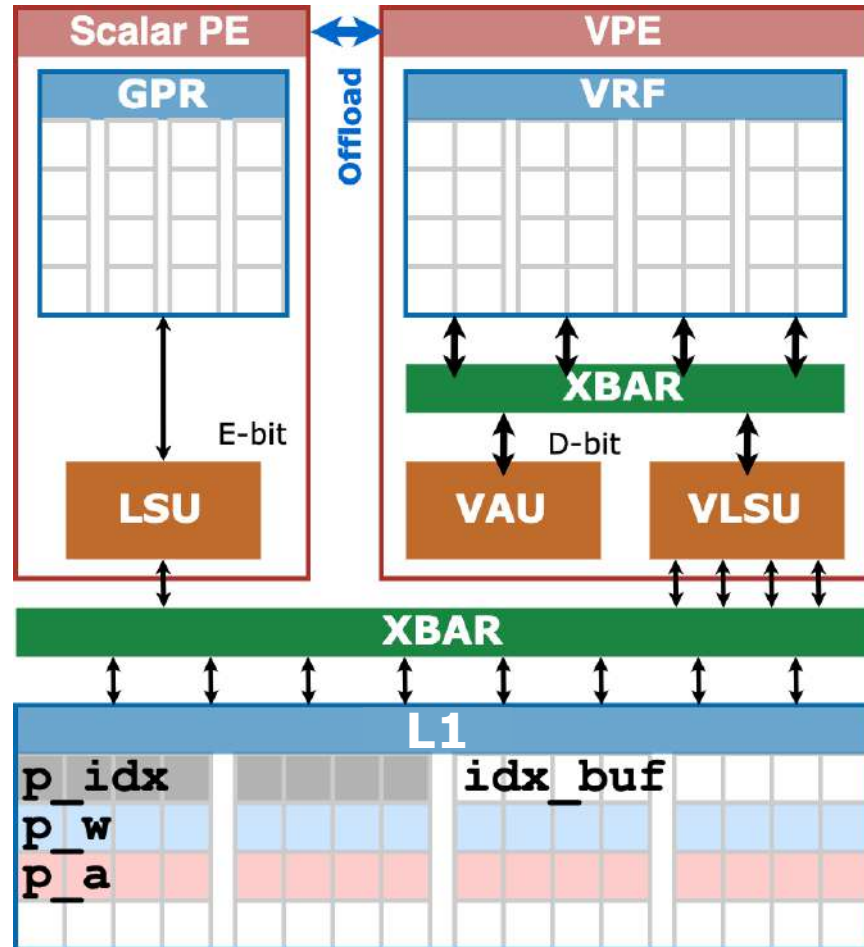
Vector Processor as the Compute Engine

- **Vector engine**
 - Computation speedup on vectorizable operations
→ **improved energy efficiency** w.r.t scalar cores
 - Compatible with vector-level operations and multi-granularity sparsity (structured and indexed)
- **Gustavson's Algorithm**
 - Natural execution model for exploiting both **adaptive activation** and **static weight** sparsity
 - Vectorized sparse tensor contraction through **metadata-driven indexed accumulation**



Index-driven Accumulation

With Conventional RISC-V Vector ISA

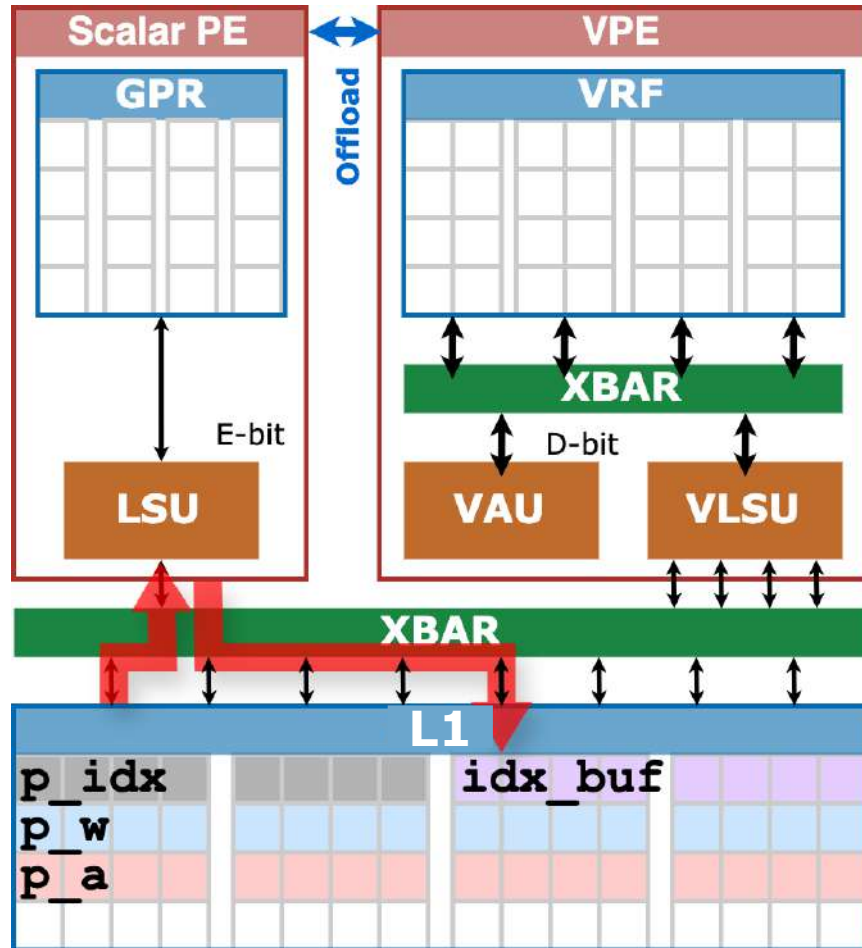


```
// pointers, settings
fp16_t    *p_w, *p_a, *p_o,
uint8_t   *p_idx, idx_buf[];
uint32_t  off_w, off_a,
vsetvli(P_tile, SEW, LMUL);
```



Index-driven Accumulation

With Conventional RISC-V Vector ISA



```
// pointers, settings
fp16_t    *p_w, *p_a, *p_o,
uint8_t   *p_idx, idx_buf[];
uint32_t   off_w, off_a,
off_meta;
vsetvli(P_tile, SEW, LMUL);
```

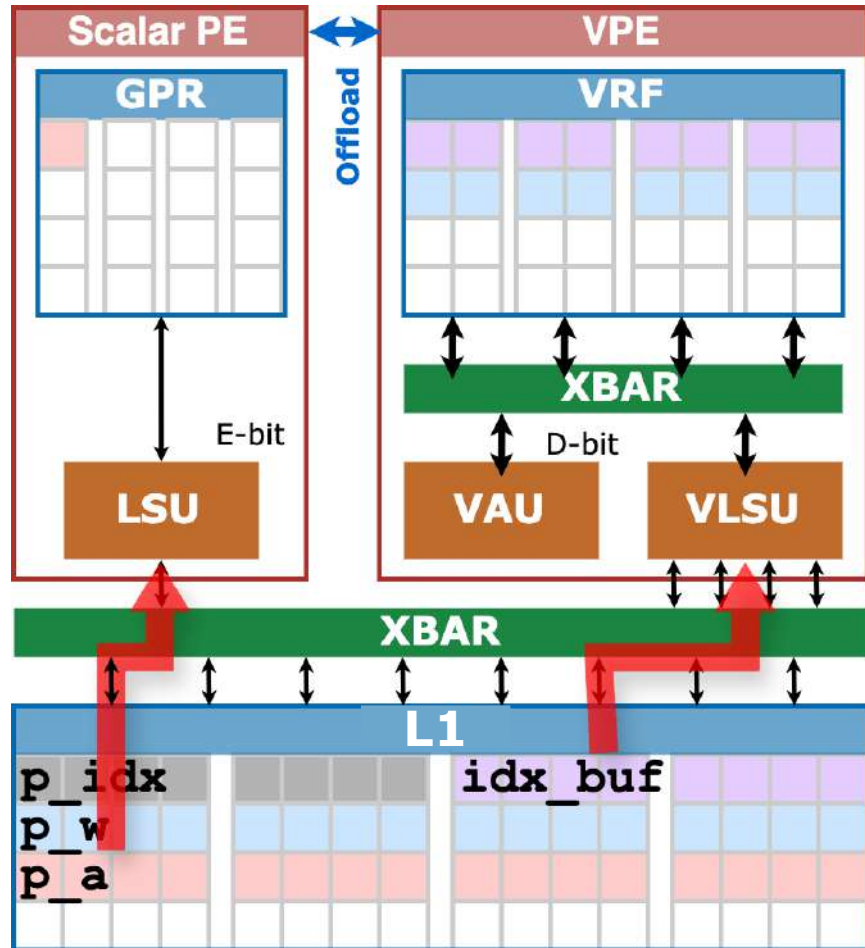
```
...
// Software decode
Loop p:
    idx_buf[p] =
    decode(p_idx[p]);
```

1

Explicit index translation

Index-driven Accumulation

With Conventional RISC-V Vector ISA



```
// pointers, settings
fp16_t  *p_w, *p_a, *p_o,
uint8_t *p_idx, idx_buf[];
uint32_t off_w, off_a,
vsetvli(P_tile, SEW, LMUL);
```

```
...
// Software decode
Loop p:
    idx_buf[p] =
    decode(p_idx[p]);

    // Load index vector
    v2 = vle8.v(idx_buf);
    // Load weight vector
    v8 = vle16.v(p_w);
    // Load scalar activation
    ft0 = flh(p_a);
    ...
```

With Conventional RISC-V Vector ISA



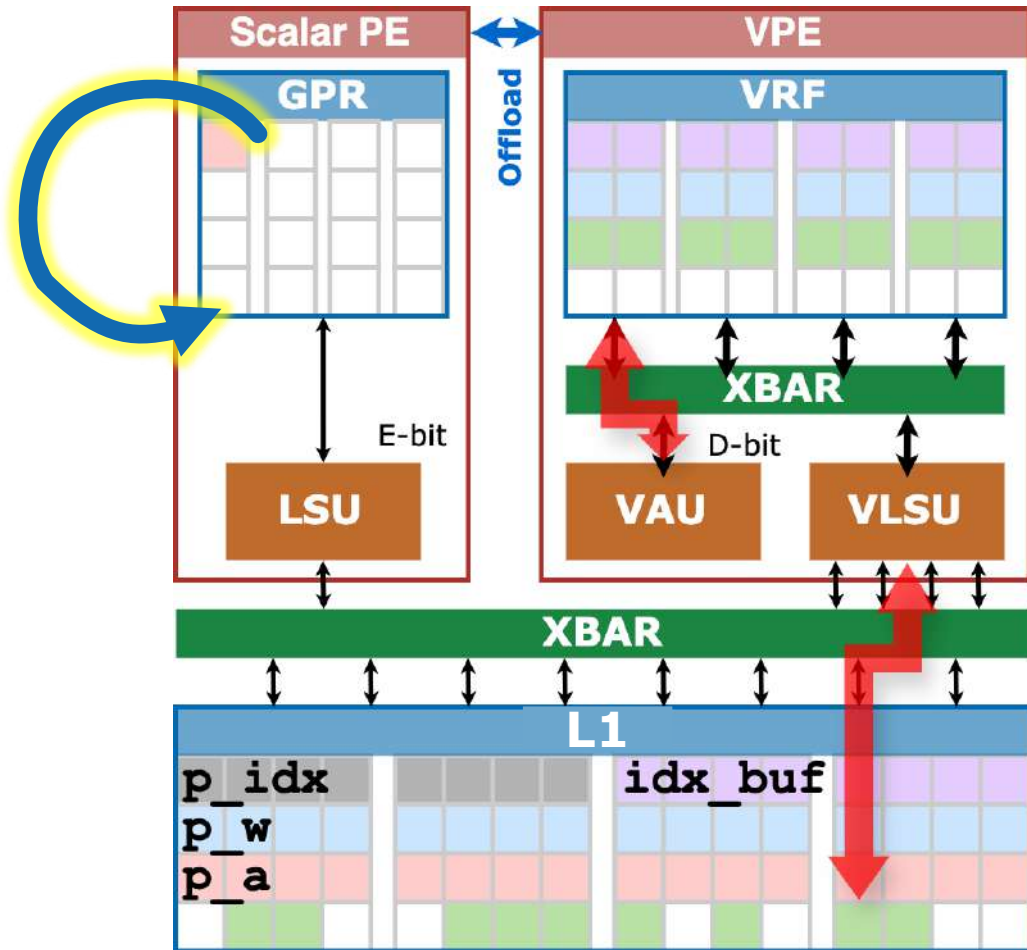
Against L1 scatter/gather

```
// Activation zero-skipping
if ft0 != 0:
    // Gather from L1
    v16 = vluxei8(p_o, v2);
    // MACC
    v16 = vfmacc.vf(v16, ft0,
v8);
    // Scatter to L1
    vsuxei8(p_o, v16, v2);
    ...
```



Index-driven Accumulation

With Conventional RISC-V Vector ISA



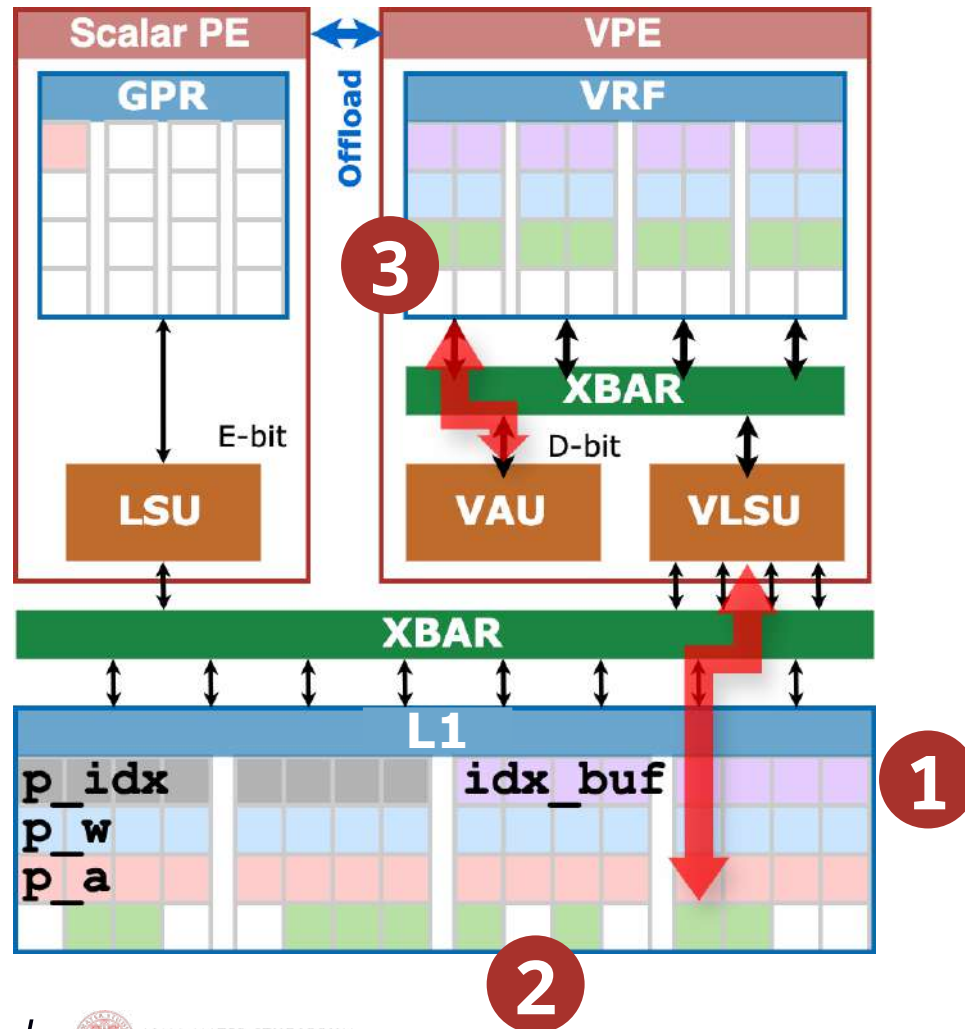
```
// pointers, settings
fp16_t    *p_w, *p_a, *p_o,
uint8_t   *p_idx, idx_buf[];
uint32_t   off_w, off_a,
vsetvli(P_tile, SEW, LMUL);
```

```
...
// Pointer updates
p_idx += off_meta;
p_w    += off_w;
p_a    += off_a;
```



- 1 *Native sparse index interpretation*
- 2 *In-core scatter/gather*
- 3 *Streamlined with arithmetic ops*

Limitation with RVV Implementations



Dedicated Index-decoding HW

General compatibility

Extra pipeline stage and L1 usage

Indexed-access to VRF

Streamlined in-core scatter/gather

VRF contention from widened accesses

VRF-resident Matrix Tile

Minimum hardware overhead

Vector-level operations inefficiency

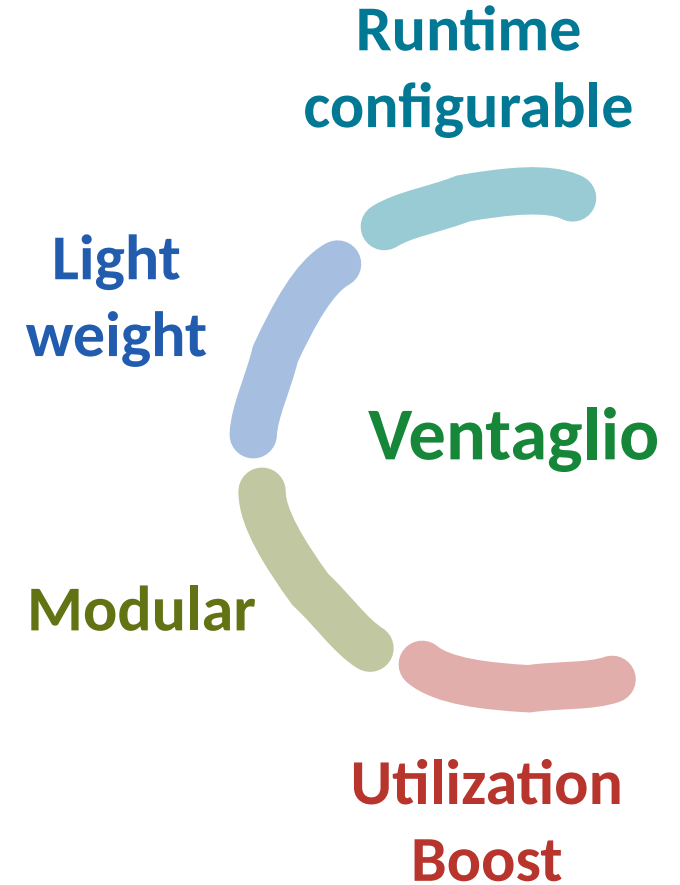
This Work - Ventaglio

Elevates index-driven accumulation to a native RVV execution primitive

Our Approach to Efficient Sparse Tensor Contraction

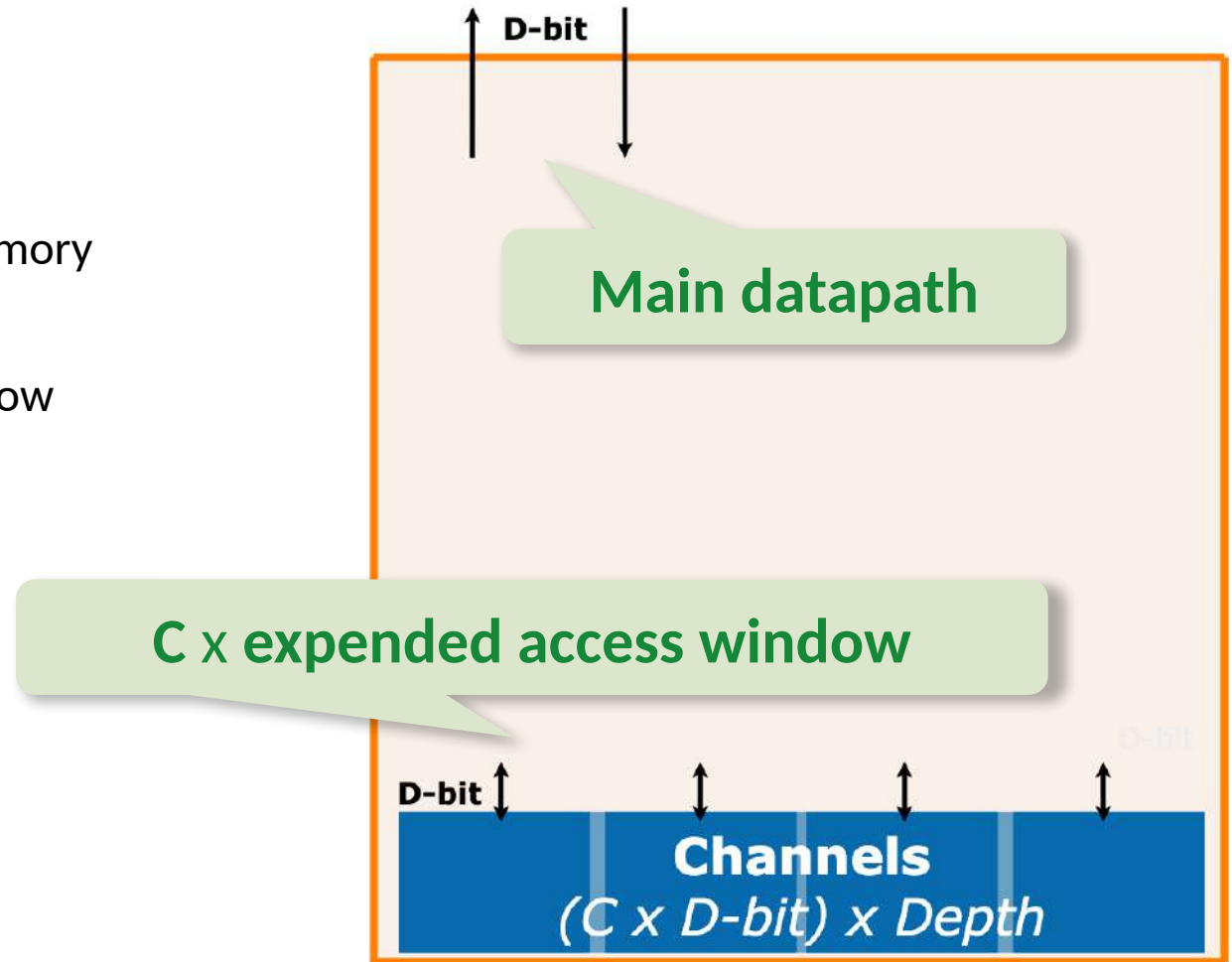


- **Ventaglio** – Elevates index-driven accumulation to a native RVV execution primitive
- Scatter/gather datapath coupled with a multi-channel memory unit
 - Modular, runtime configurable
- System integration
 - Dual-core Spatz Cluster, 256 KiB L1 SPM
 - Light-weight, **3.1%** area overhead
 - Scale-out evaluation on a 4x4 multi-cluster system
- Evaluation
 - **At-the-roofline efficiency for sparse tensor contractions**
 - **6.9–7.4x** speedups over the optimized RVV kernel baselines
 - DuoGPT–pruned LLaMA-3-8B model: **2.06–5.25x** speedup



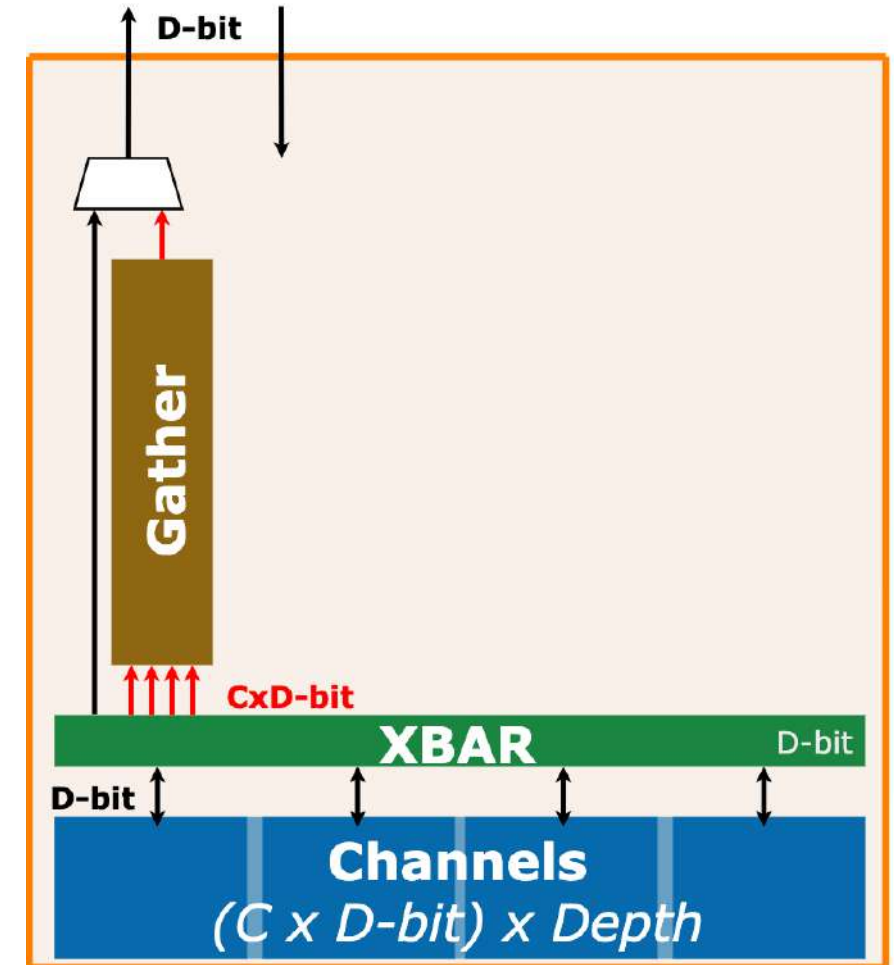
Ventaglio Architecture

- **Goal:** Materializes only the referenced lanes required by each sparse update
- **Memory unit:**
 - A multichannel latch-based standard-cell memory
 - Maintains expanded data structures
 - Provides a contiguous expanded access window



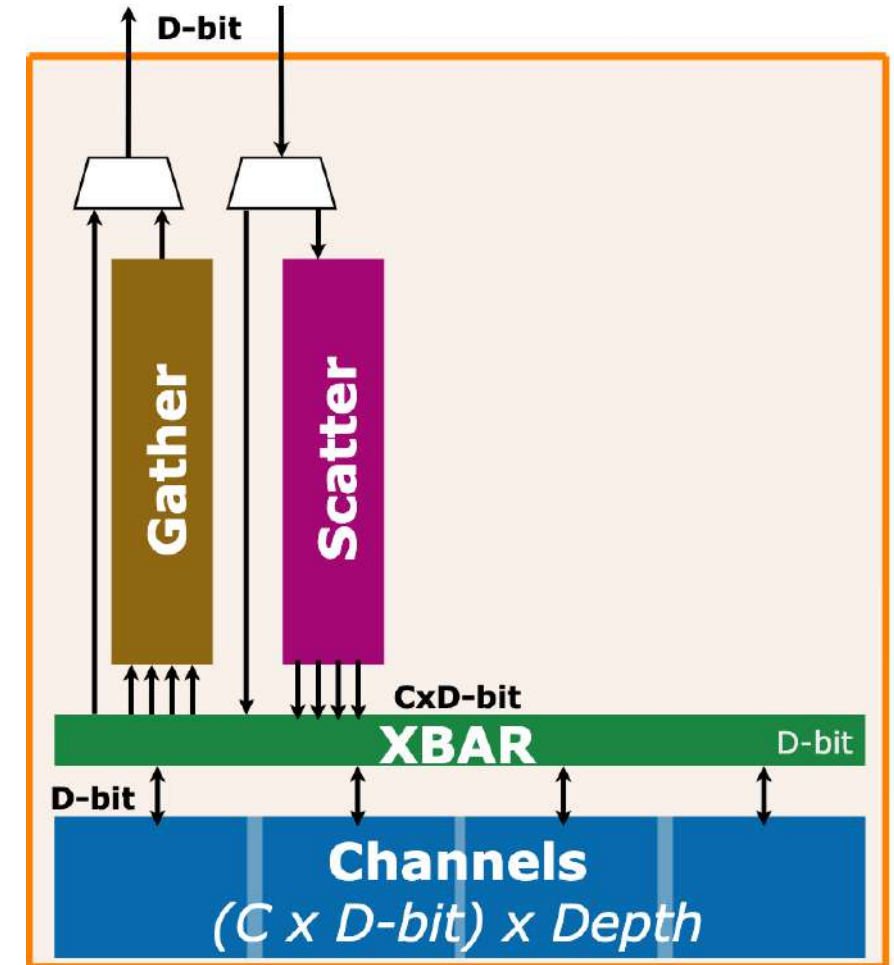
Ventaglio Architecture

- **Goal:** Materializes only the referenced lanes required by each sparse update
- **Memory unit:**
 - A multichannel latch-based standard-cell memory
 - Maintains expanded data structures
 - Provides a contiguous expanded access window
- **Datapaths:**
 - Gather datapath converts wide, multi-channel reads $\rightarrow$ compact operand representation



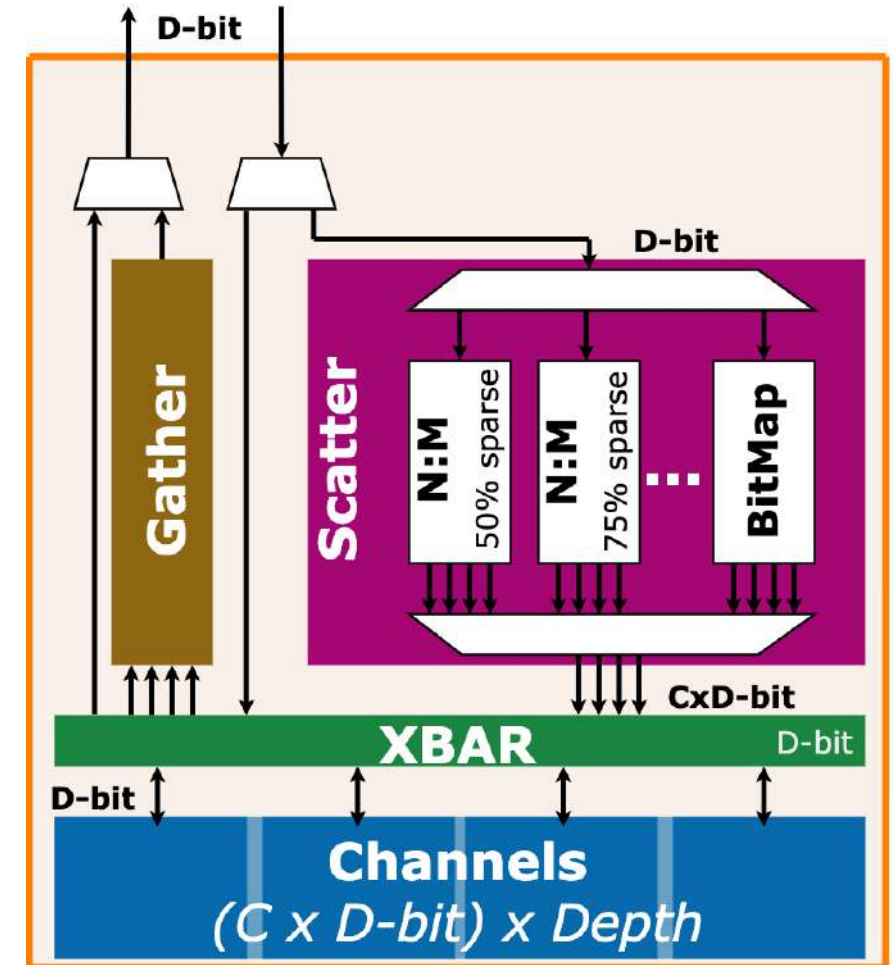
Ventaglio Architecture

- **Goal:** Materializes only the referenced lanes required by each sparse update
- **Memory unit:**
 - A multichannel latch-based standard-cell memory
 - Maintains expanded data structures
 - Provides a contiguous expanded access window
- **Datapaths:**
 - Gather datapath converts wide, multi-channel reads $\rightarrow$ compact operand representation
 - Scatter datapath performs the inverse mapping for write-back



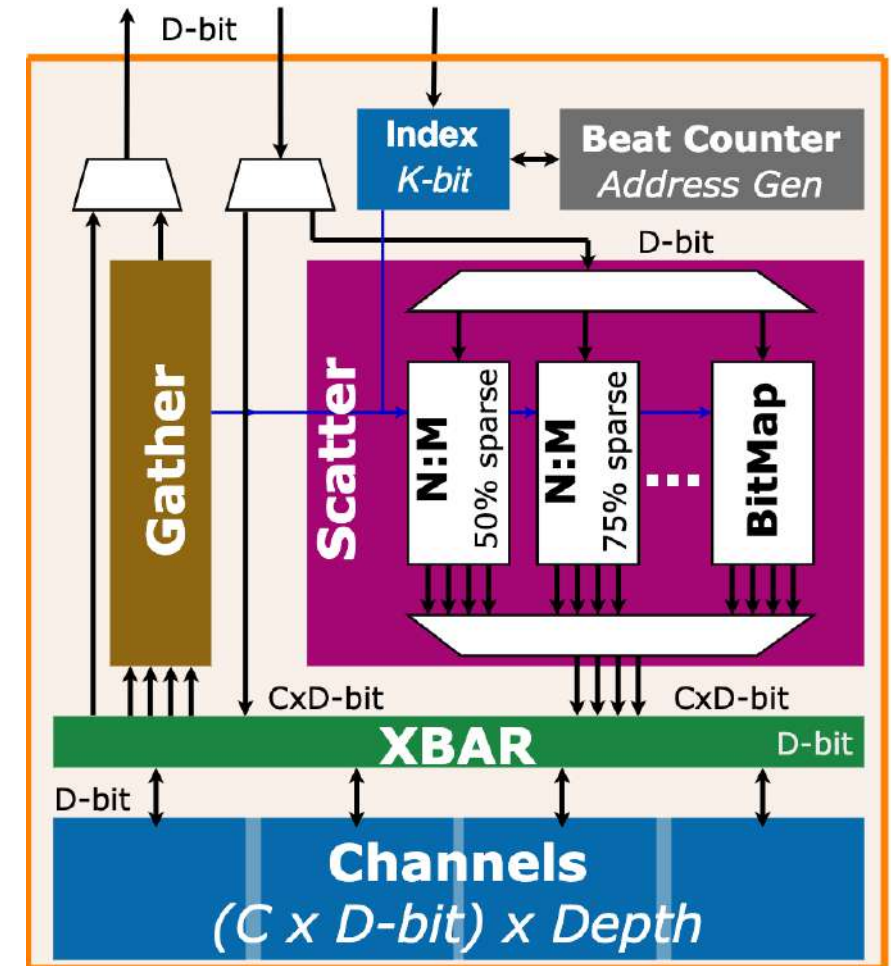
Ventaglio Architecture

- **Goal:** Materializes only the referenced lanes required by each sparse update
- **Memory unit:**
 - A multichannel latch-based standard-cell memory
 - Maintains expanded data structures
 - Provides a contiguous expanded access window
- **Datapaths:**
 - Gather datapath converts wide, multi-channel reads $\rightarrow$ compact operand representation
 - Scatter datapath performs the inverse mapping for write-back
 - Modular datapaths, with only one sparsity mode active at a time



Ventaglio Architecture

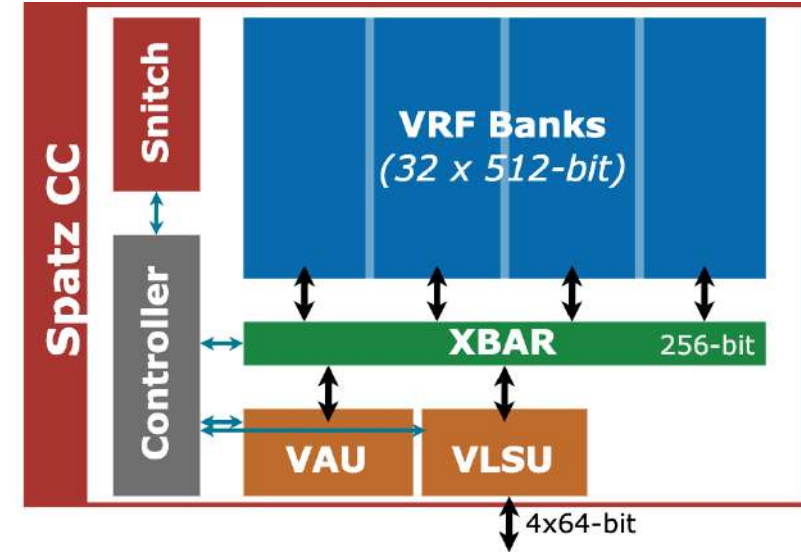
- **Goal:** Materializes only the referenced lanes required by each sparse update
- **Memory unit:**
 - A multichannel latch-based standard-cell memory
 - Maintains expanded data structures
 - Provides a contiguous expanded access window
- **Datapaths:**
 - Gather datapath converts wide, multi-channel reads $\rightarrow$ compact operand representation
 - Scatter datapath performs the inverse mapping for write-back
 - Modular datapaths, with only one sparsity mode active at a time
- **Control logics:**
 - Packed metadata consumption & preload



Spatz Cluster: Ventaglio Integration

- **Spatz Core-Complex (CC):**

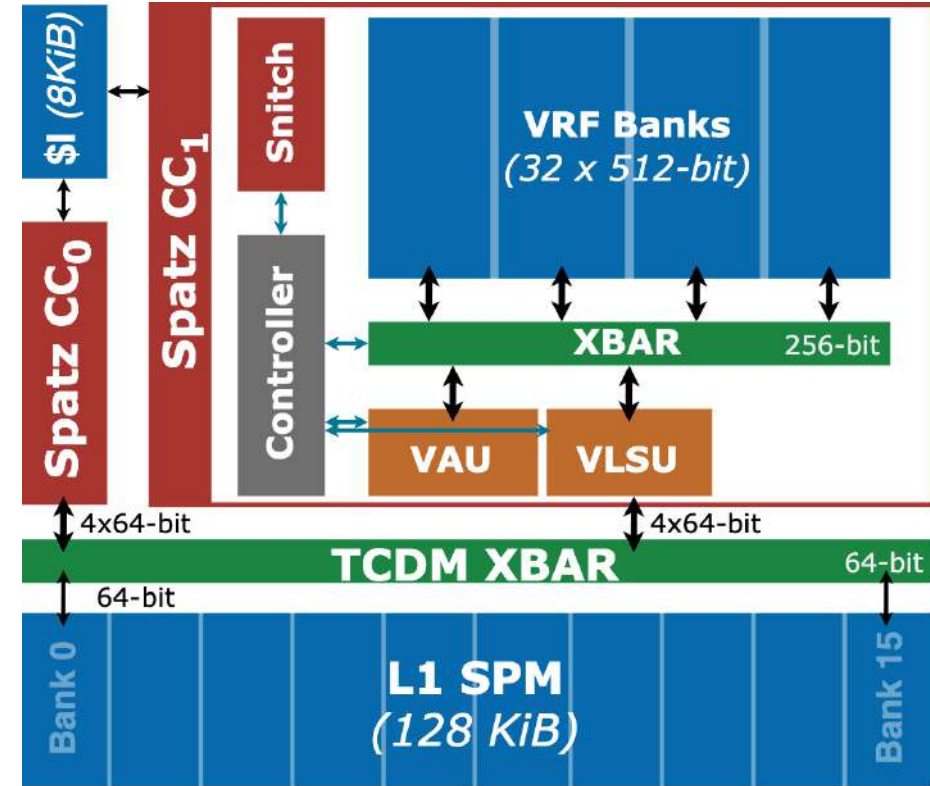
- Spatz VPE tightly coupled to a tiny single-stage RV32I Snitch core
- VPE comprises a controller, a latch-based VRF, and parallel functional units



Spatz Cluster: Ventaglio Integration



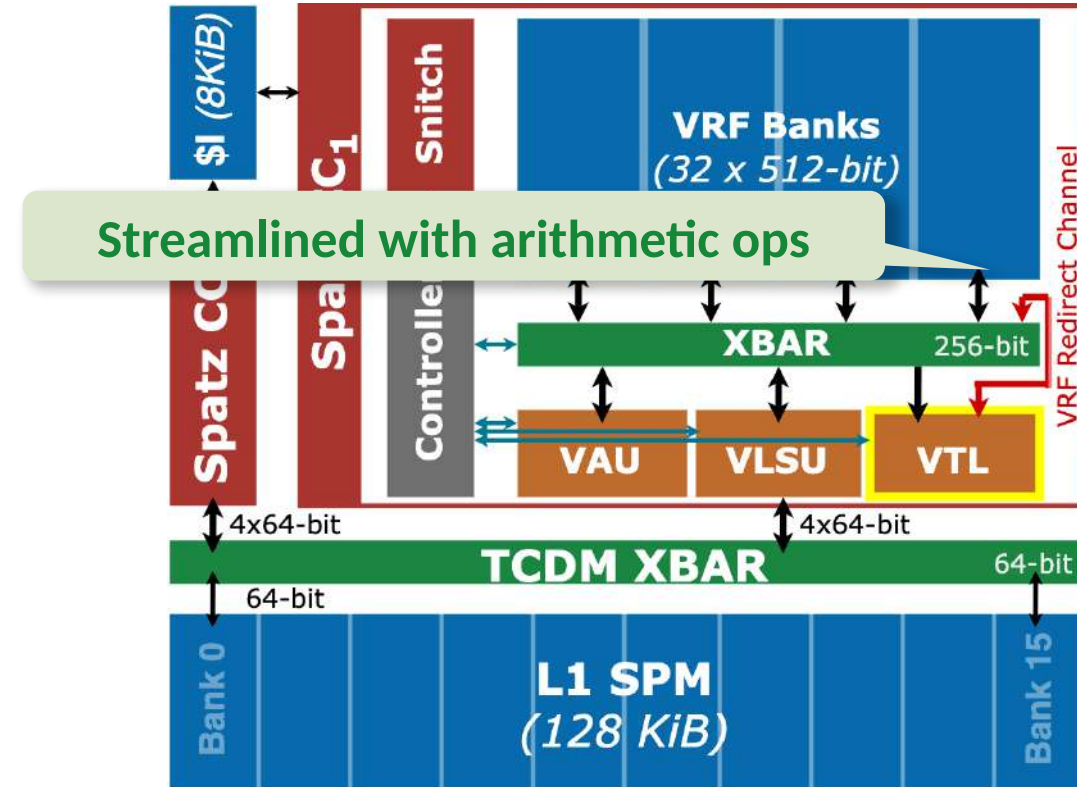
- **Spatz Core-Complex (CC):**
 - Spatz VPE tightly coupled to a tiny single-stage RV32I Snitch core
 - VPE comprises a controller, a latch-based VRF, and parallel functional units
- **Spatz Cluster:**
 - Two Spatz CCs share a 128-KiB L1 Scratchpad Memory (SPM)
 - Logarithmic crossbar with single-cycle latency



Spatz Cluster: Ventaglio Integration



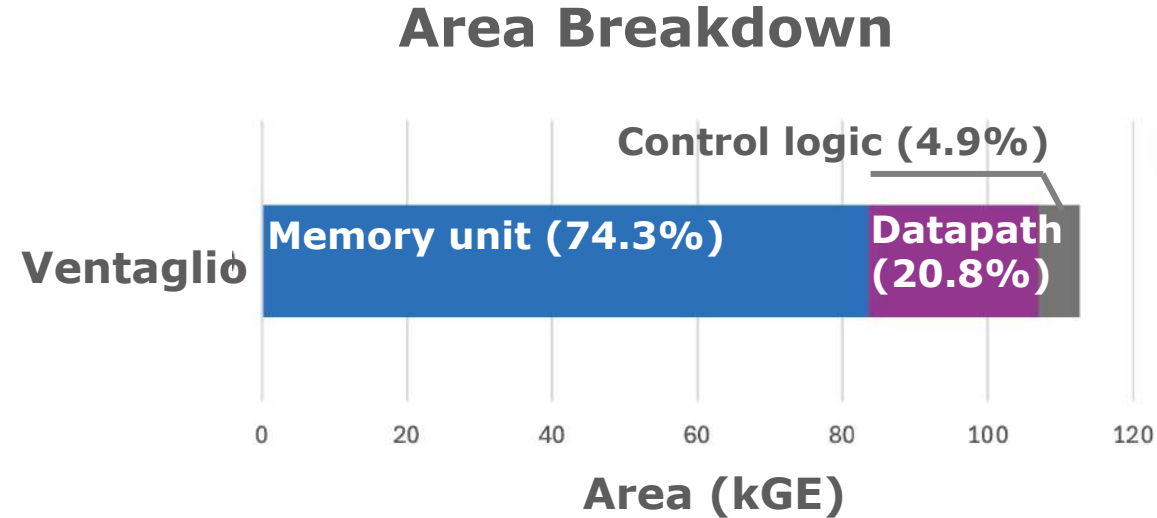
- **Spatz Core-Complex (CC):**
 - Spatz VPE tightly coupled to a tiny single-stage RV32I Snitch core
 - VPE comprises a controller, a latch-based VRF, and parallel functional units
- **Spatz Cluster:**
 - Two Spatz CCs share a 128-KiB L1 Scratchpad Memory (SPM)
 - Logarithmic crossbar with single-cycle latency
- **Ventaglio: a configurable sparse execution unit**
 - Active datapath is selected through Control and Status Registers (CSRs) in the controller
 - **Dynamically mapping** selected vector registers to its memory unit
 - Redirects VRF accesses through a **bypass channel**



Results: Area Breakdown



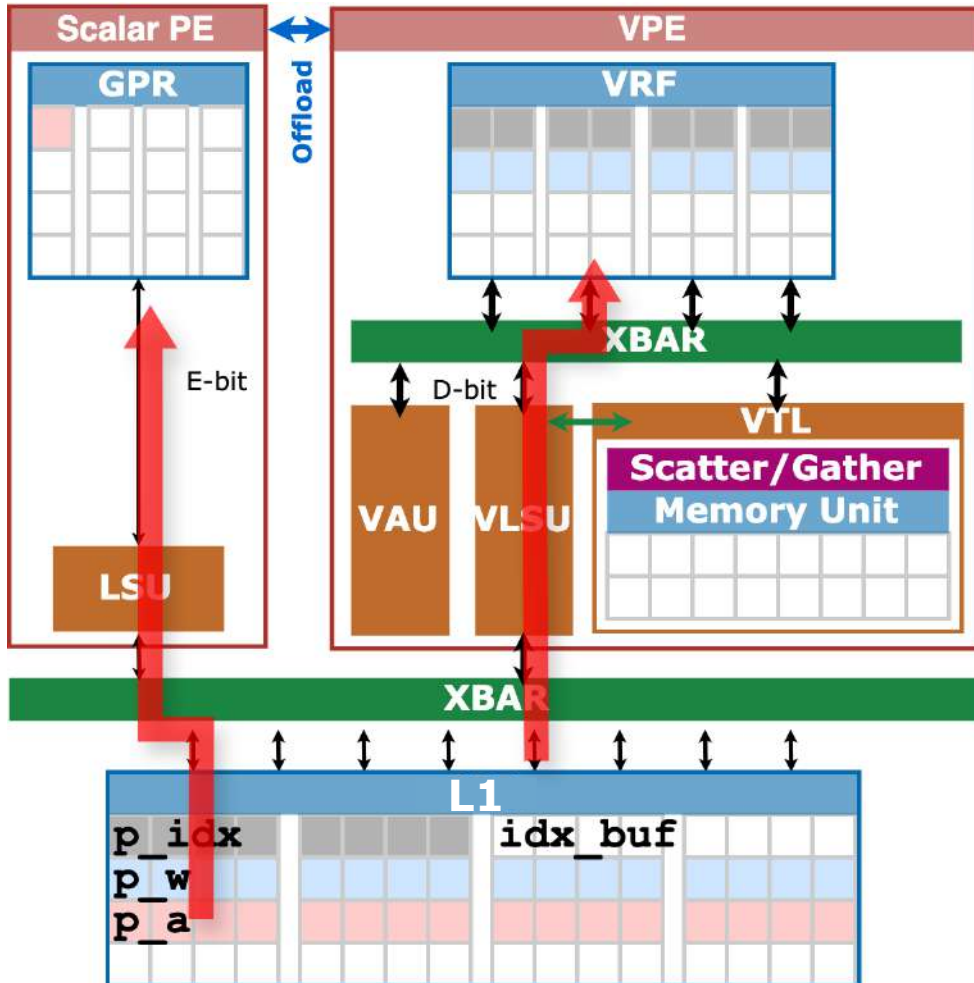
- Ventaglio is implemented with **GlobalFoundries' 12LPPLUS FinFET technology**
 - Instantiates 1 KiB memory unit
 - Supports 1:4, 2:4, and bitmap sparse formats
- Maximum frequency: 1 GHz (TT/0.8V/25°C)
 - Ventaglio **does not** impact critical timing path
- Cluster area overhead: **112.7 kGE**



3.1% Increment

Index-driven Accumulation

With Ventaglio Extension



```
// pointers, settings
fp16_t  *p_w, *p_a, *p_o,
uint8_t  *p_idx, idx_buf[];
uint32_t off_w, off_a,
vsetvli(P_tile, SEW, LMUL);
```

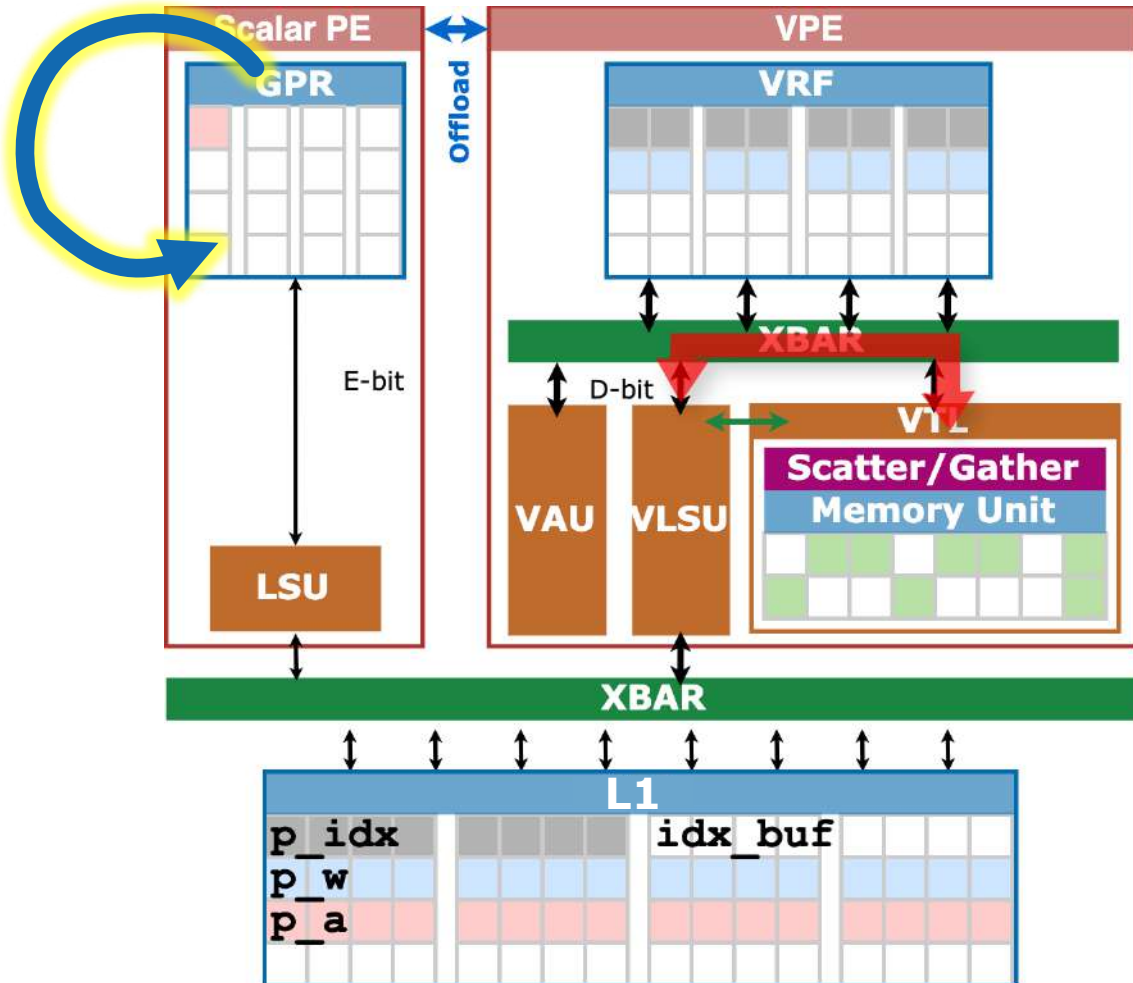
```
// Ventaglio config
cfg_vtl(v16, sp_fmt);
```

```
...
// Load index vector
v2  = vlx.v(p_idx);
// Load weight vector
v8  = vle16.v(p_w);
// Load scalar activation
ft0 = flh(p_a);
...
```



Index-driven Accumulation

With Ventaglio Extension



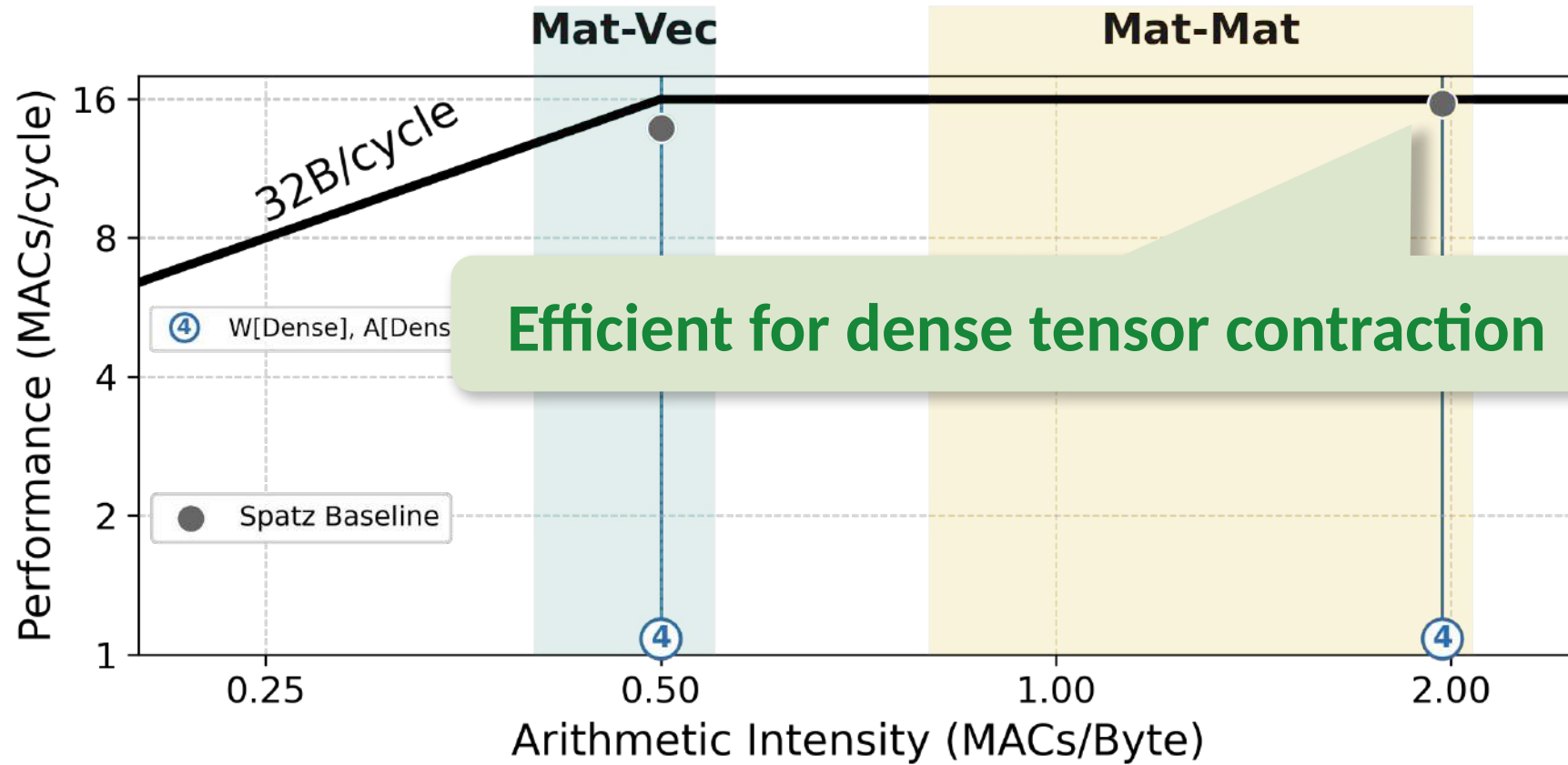
```
// pointers, settings
fp16_t    *p_w, *p_a, *p_o,
uint8_t   *p_idx, idx_buf[];
uint32_t   off_w, off_a,
off_meta;
vsetvli(P_tile, SEW, LMUL);
```

```
// Ventaglio config
cfg_vtl(v16, sp_fmt);
```

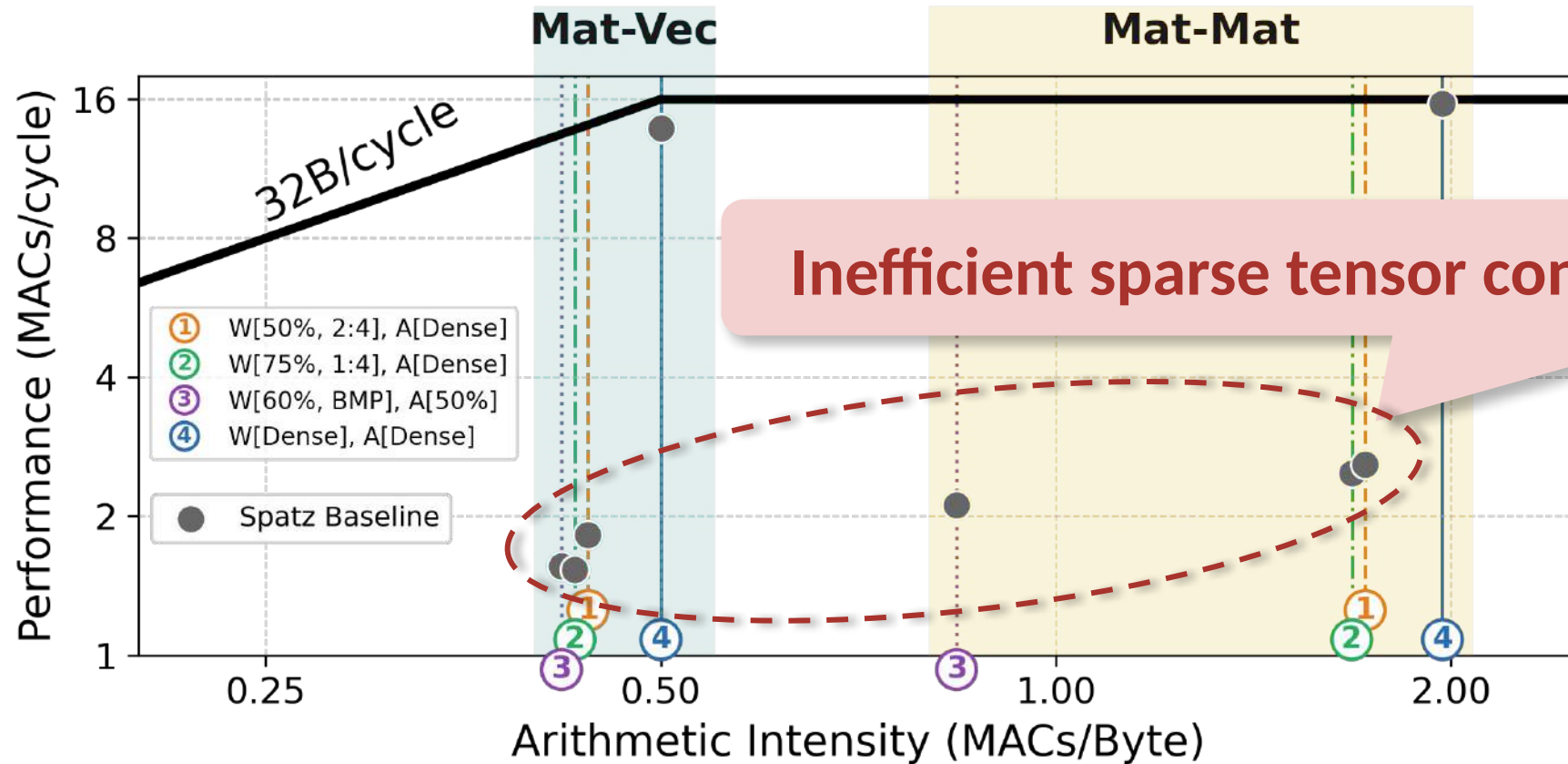
```
...
// Activation zero-skipping
if ft0 != 0:
    // MACC
    v16 = vfxmacc.vf(v16, ft0
                     v8, v2);
```

```
// Pointer update
p_idx += off_meta;
p_w   += off_w;
p_a   += off_a;
```

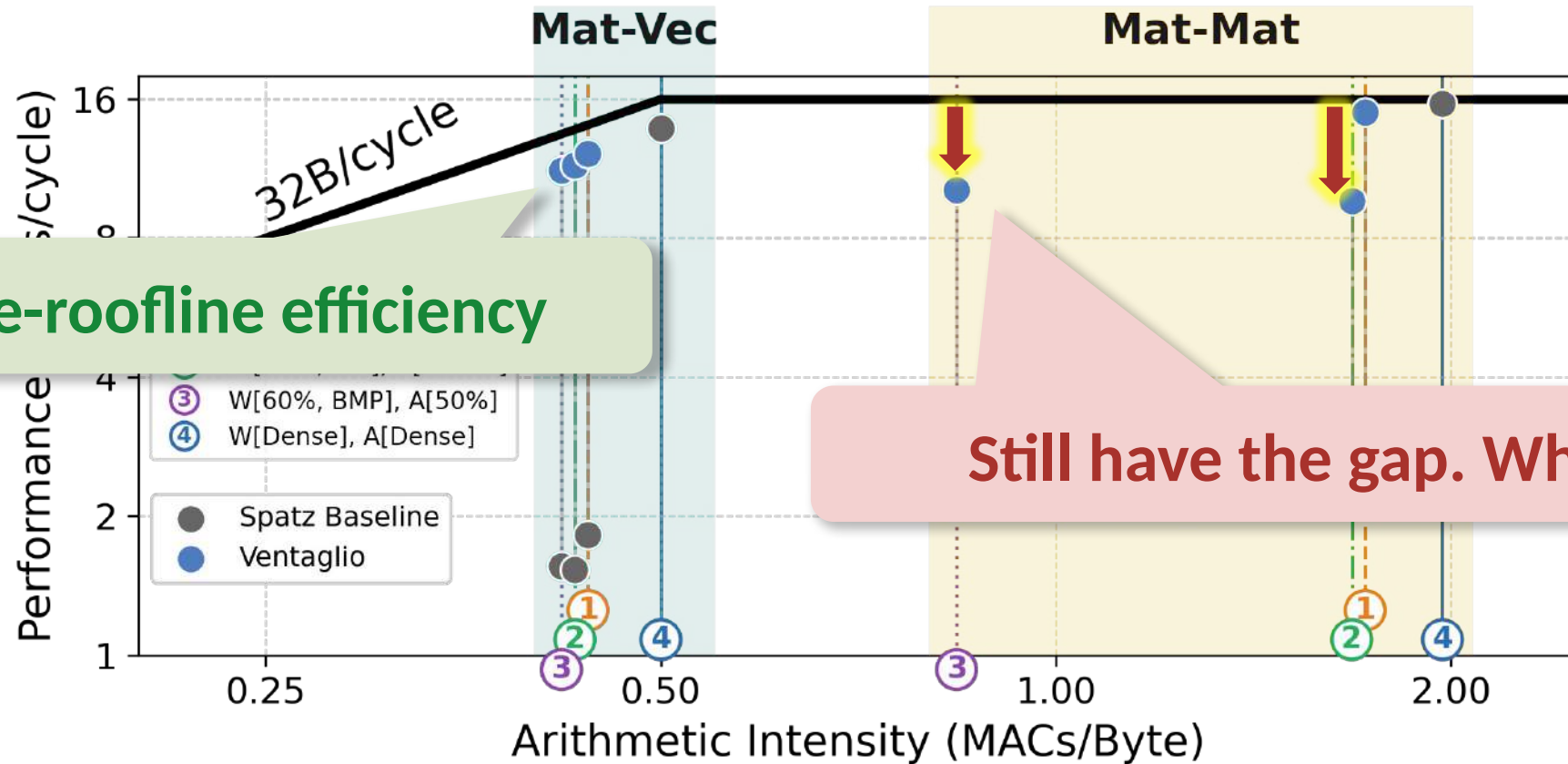
Results: Sparse Tensor Contraction



Results: Sparse Tensor Contraction

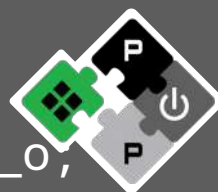
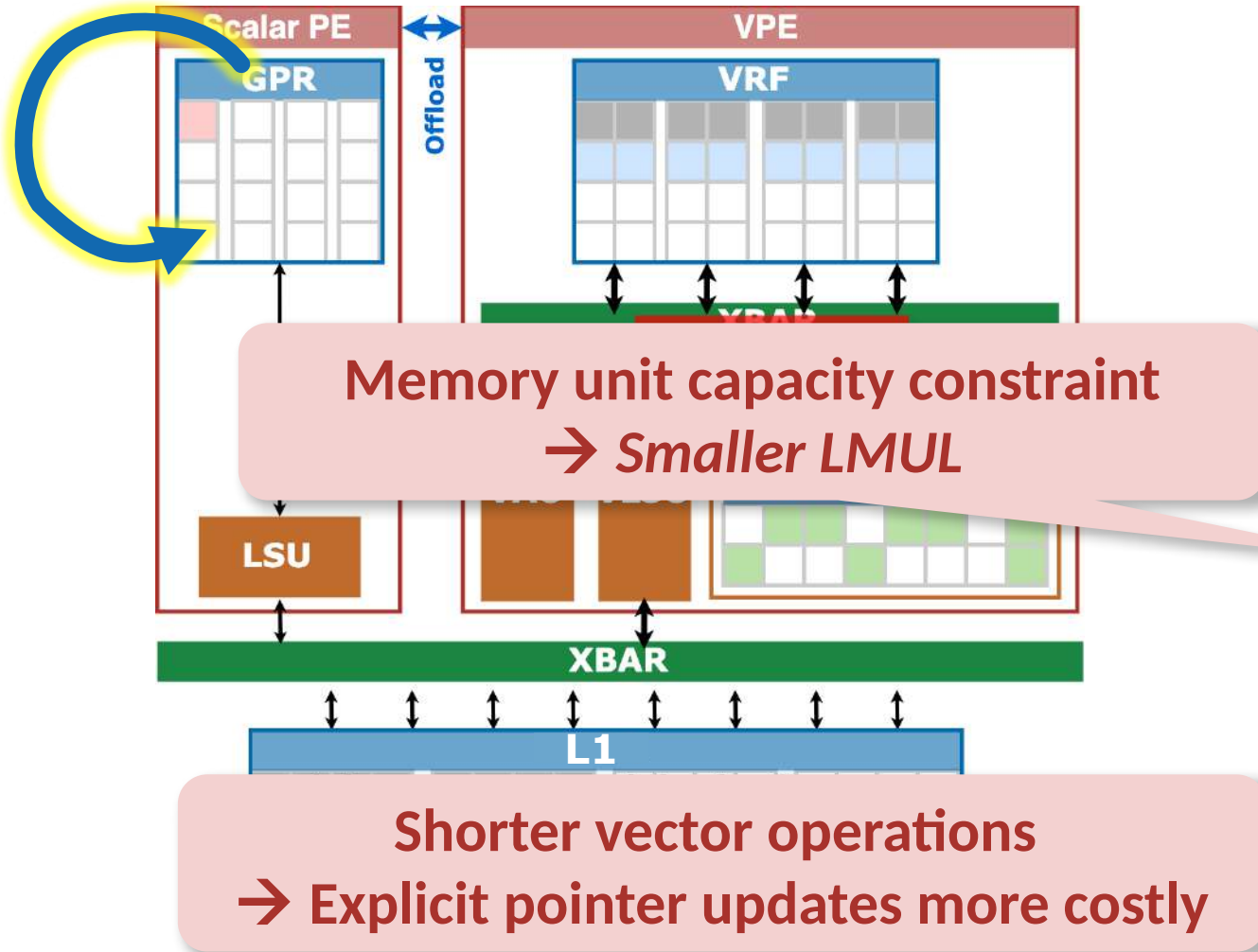


Results: Sparse Tensor Contraction



Index-driven Accumulation

With Ventaglio Extension



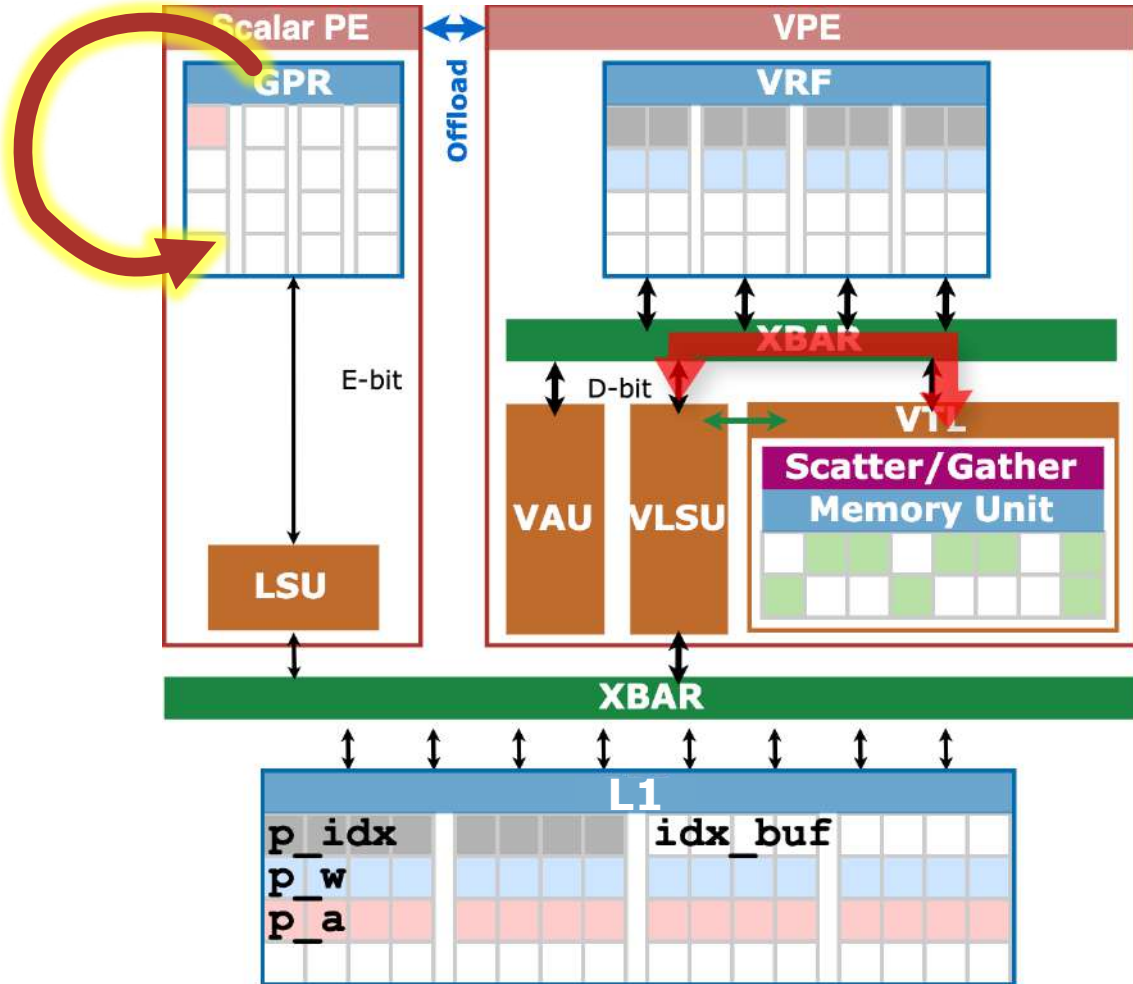
```
// pointers, settings
fp16_t  *p_w, *p_a, *p_o,
uint8_t *p_idx, idx_buf[];
uint32_t off_w, off_a,
off_meta;
vsetvli(P_tile, SEW, LMUL);
```

```
// Ventaglio config
cfg_vt1(v16, sp_fmt);
```

```
...
// Activation zero-skipping
if ft0 != 0:
    // MACC
    v16 = vfxmacc.vf(v16, ft0
                    v8, -v2);
// Pointer update
p_idx += off_meta;
p_w   += off_w;
p_a   += off_a;
```

Index-driven Accumulation

With Ventaglio Extension & Address Post-increment



```
// pointers, settings
fp16_t    *p_w, *p_a, *p_o;
uint8_t    *p_idx, idx_buf[];
uint32_t    off_w, off_a,
off_meta;
vsetvli(P_tile, SEW, LMUL);
```

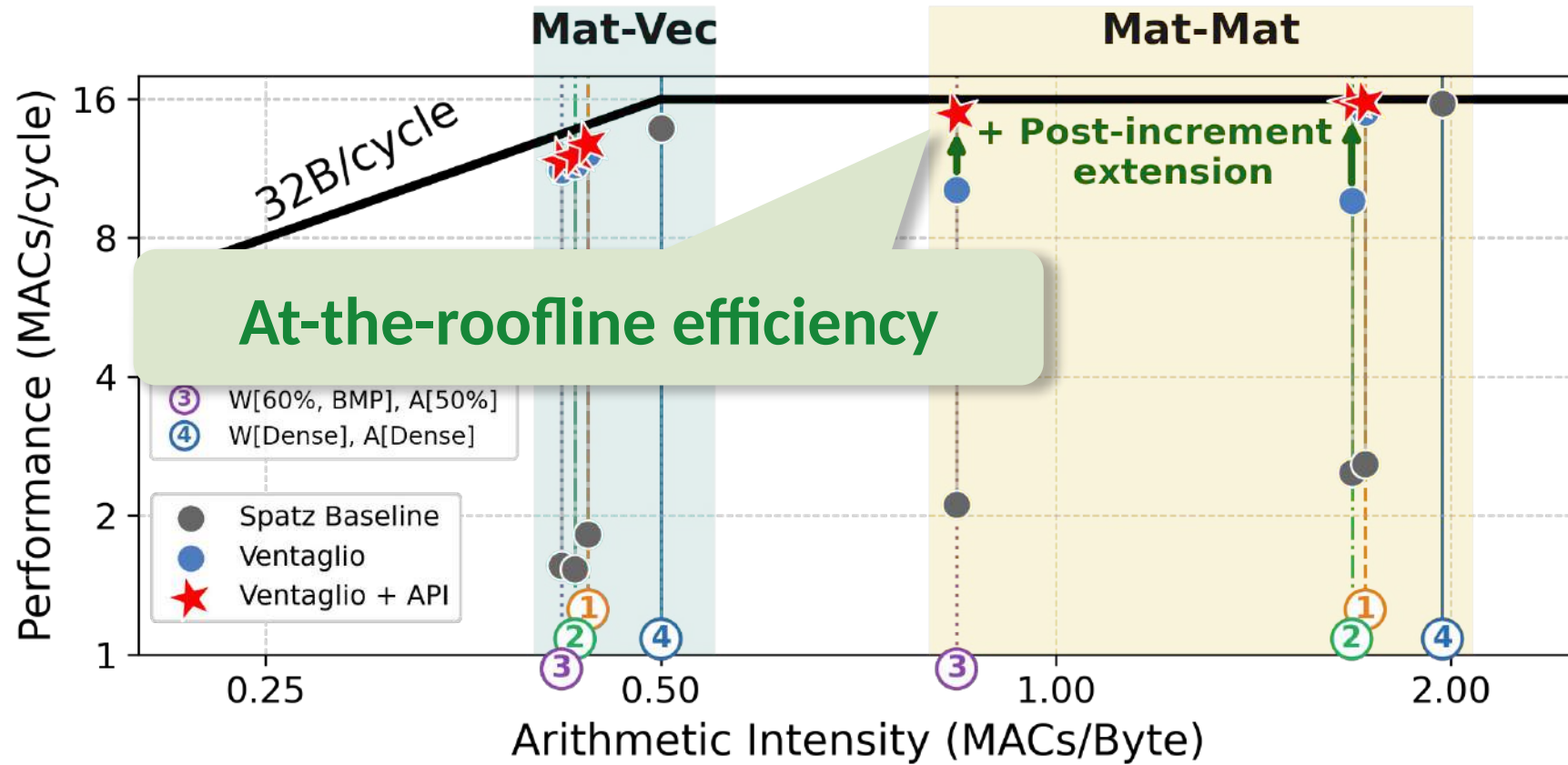
```
// Ventaglio config
cfg_vt1(v16, sp_fmt);
```

```
...
// Load index, weight, acti
v2  = p.v1x.v(p_idx, off_idx);
v8  = p.v1e16.v(p_w, off_w);
ft0 = p.flh(p_a, off_a);
```

```
// Activation zero-skipping
if ft0 != 0:
    // MACC
    v16 = vfxmacc.vf(v16, ft0,
                     v8, v2);
```



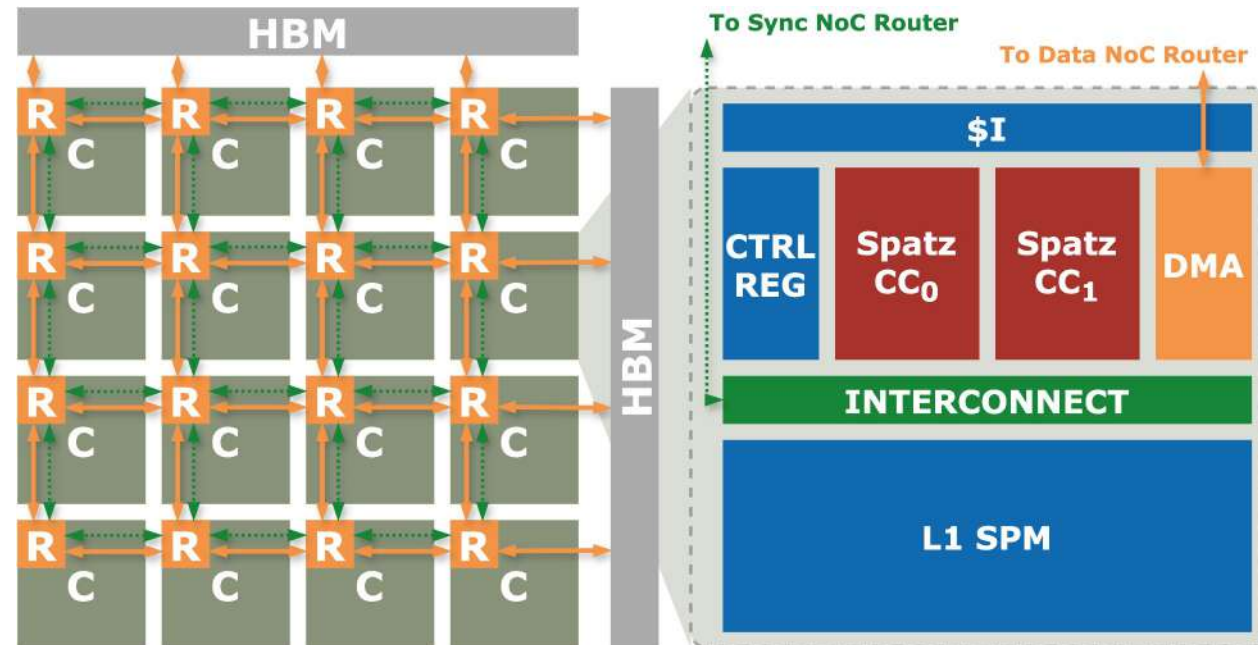
Results: Sparse Tensor Contraction



Results: End-to-End Evaluation



- Performance-accurate instruction-level Ventaglio model in GVSoC
 - Scale to a 4×4 multi-cluster system connected through a 2D NoC mesh
 - Per-cluster 1024-bit network link + row and column-wise **multicast** and **sum reduction**
 - Off-chip memory modeled in DRAMSys as 4 × 2 HBM2 channels (agg. 512 GiB/s)
- **Workload**
 - *DuoGPT-pruned* LLaMA3-8B model with practical **40–60% dual-sparsity**



Results: End-to-End Evaluation

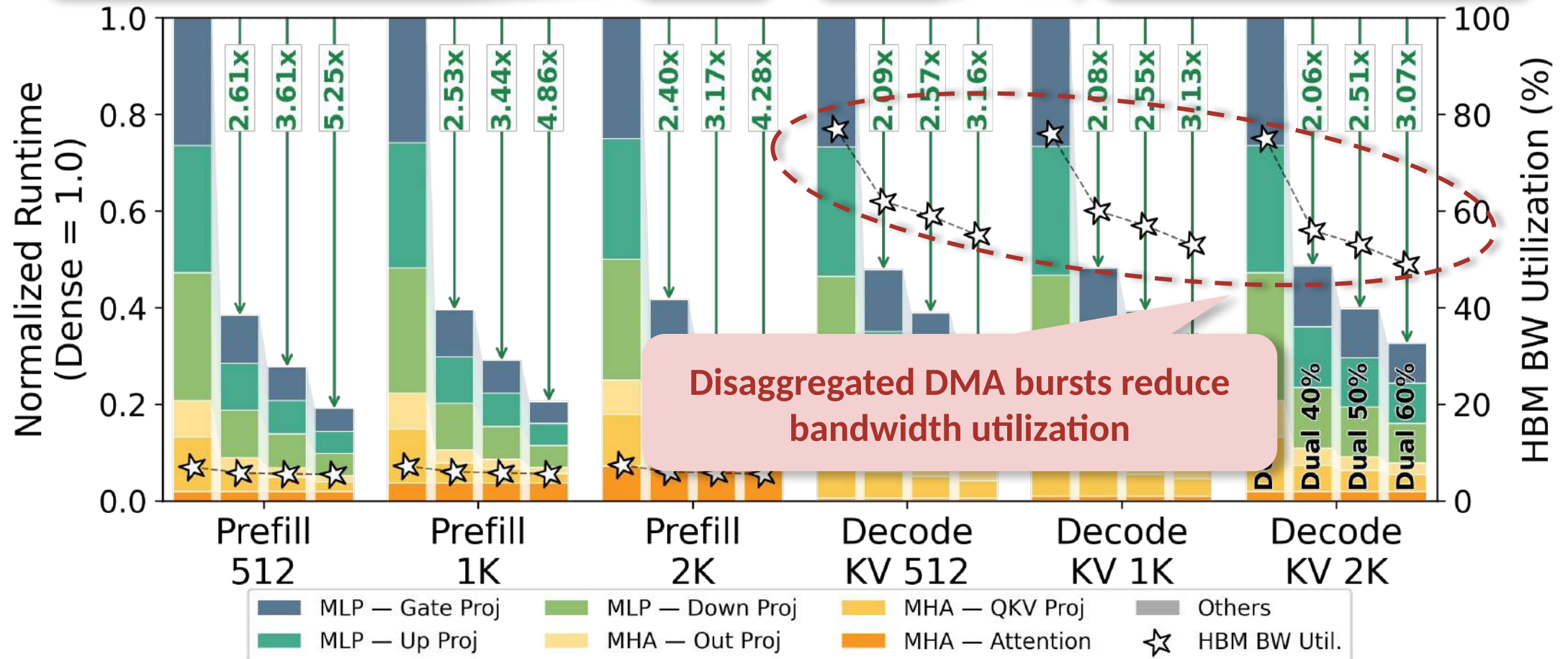


Compute-bound

Matrix-matrix projections dominate execution

Memory-bound

Matrix-vector projections load corresponding weight tile per cluster



Ventaglio: Low-overhead HW + ISA = Efficiency Boost



- Elevating index-driven accumulation to native RVV primitives
- **Lightweight**
 - 3.1% logic area overhead
- **Flexible, runtime programmable**
 - Modular extension for other sparsity formats
- **Utilization Boost**
 - 2.06-5.25x speedup in real-world end-to-end applications
- **Work in progress**
 - HW-SW-Format co-design for extreme sparse ratio

[1] NVIDIA. "NVIDIA Hopper Architecture In Depth," Available: <https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth>

[2] V. Titopoulos et al., "Optimizing structured-sparse matrix multiplication in RISC-V vector processors," IEEE Transactions on Computers, vol. 74, no. 4, pp. 1446–1460, 2025.

[3] X. Tengfei et al., "RISC-V ISA Extensions for Vectorized Unstructured Sparse SpMM in LLM Inference," 2026 Design, Automation & Test in Europe Conference & Exhibition. IEEE.



github.com/pulp-platform/spatz