ETH zürich

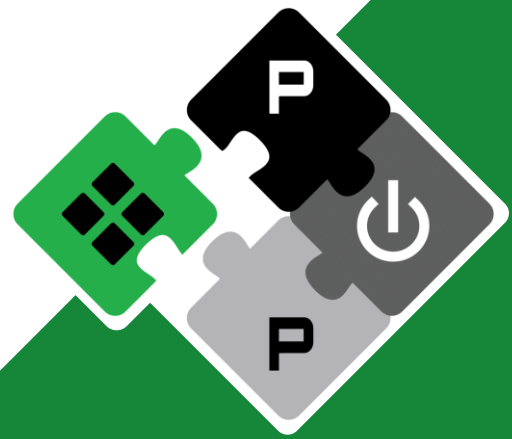
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

TROOP: At-the-Roofline Performance for Vector Processors on Low Operational Intensity Workloads

Integrated Systems Laboratory (ETH Zürich)

Navaneeth Kunhi Purayil nkunhi@iis.ee.ethz.ch
Diyou Shen dishen@iis.ee.ethz.ch
Dr. Matteo Perotti mperotti@iis.ee.ethz.ch

Prof. Dr. Luca Benini lbenini@iis.ee.ethz.ch

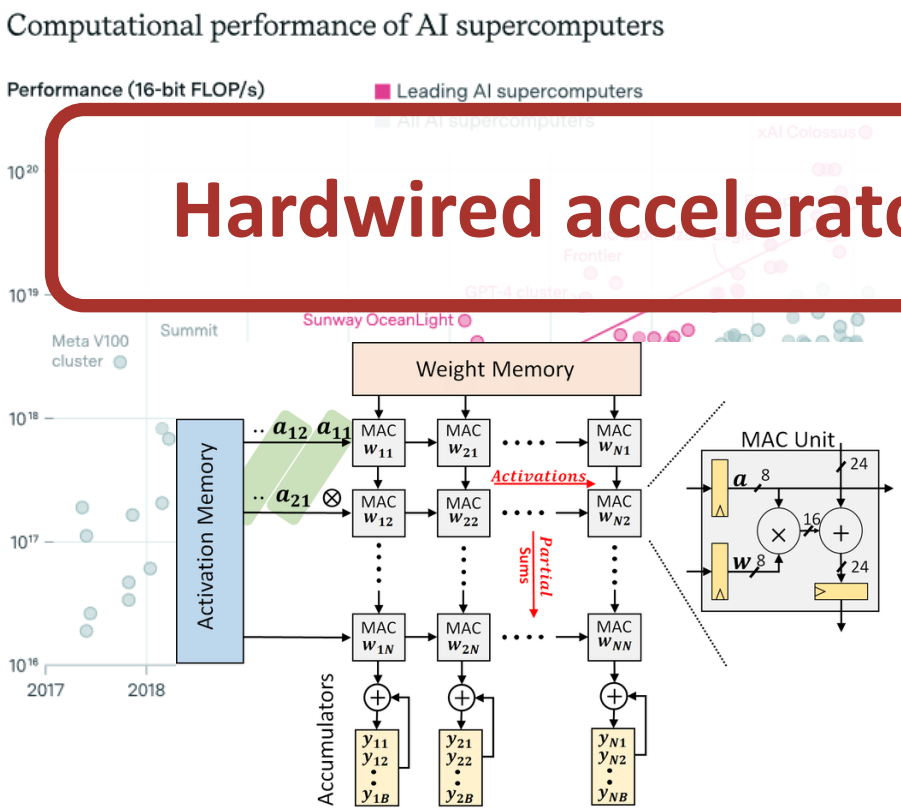


PULP Platform
Open Source Hardware, the way it should be!

pulp-platform.org
@pulp_platform
company/pulp-platform
youtube.com/pulp_platform

Fast changing pace of AI

- **Ever increasing model complexity**
  - Beyond exascale AI supercomputers!



Dataflow

Fast algorithmic evolution!

DNNs/CNNs/LLMs (Compute-intensive)

Generative LLMs/MoE (Memory-intensive)

Hardwired accelerators suffer from rapid obsolescence!

Programmable many-core SIMD/SIMT

Resurgence of Vector Processors

- **Vector Processor Features**
  1. Exploit Data parallelism
  2. Simple Programming model
  3. Software Portability
  4. **High Flexibility**

arm

NEC

RISC-V

RISC-V Highly Flexible → V 1.0 in 2021

Datacenter

Edge Computing

SiFive

semidynamics

openchip

Automotive

Defense

VENTANA

tenstorrent

ANDES TECHNOLOGY

T-HEAD

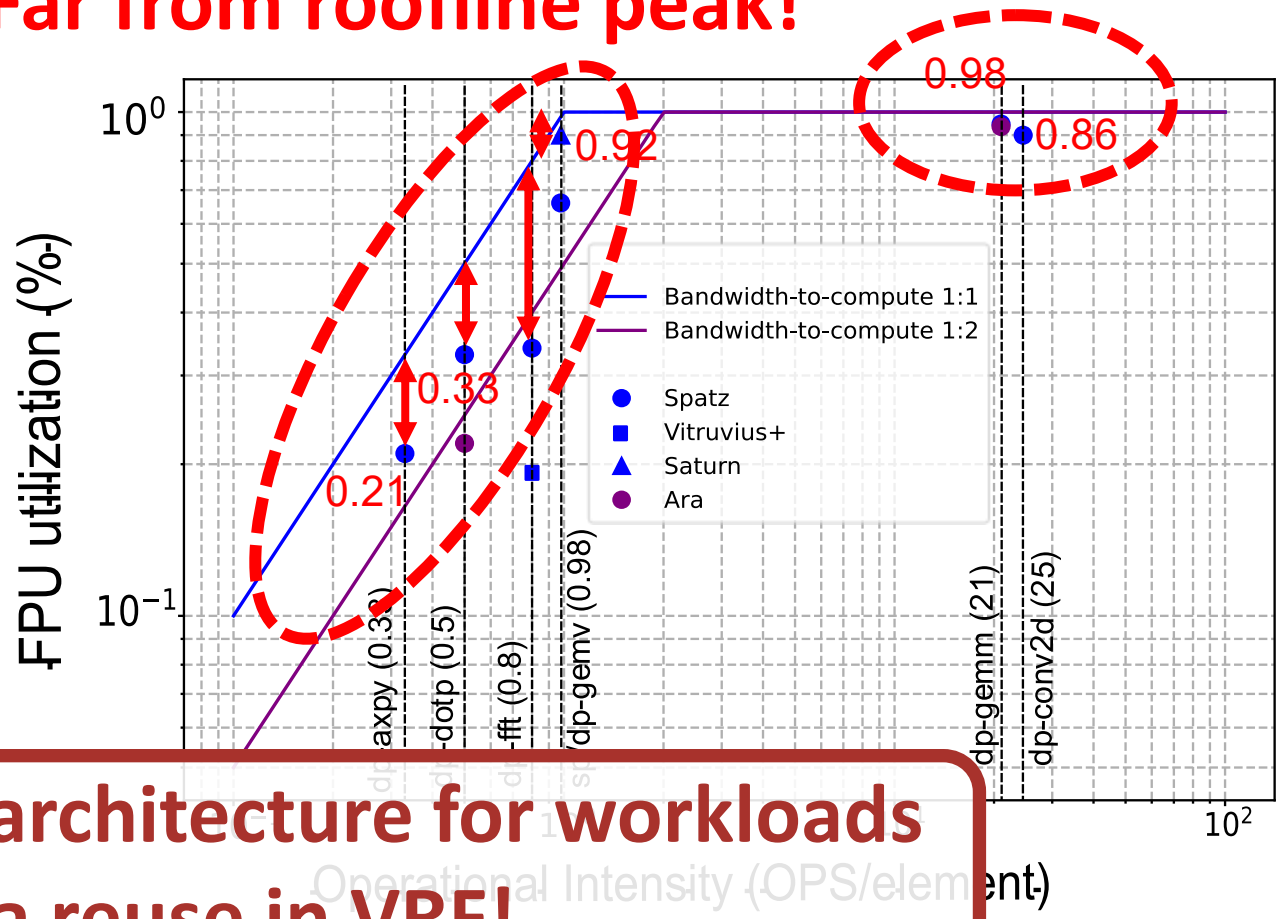
SoA VPEs: Low efficiency for memory bound workloads

Challenge:

✔ **SoA VPEs optimized for compute-intensive workloads**

✘ **Achieve poor FPU utilizations far from roofline for memory intensive workloads**

Far from roofline peak!



Need to optimize micro-architecture for workloads with low-data reuse in VRF!

SoA VPEs: Low efficiency for memory bound workloads

We propose TROOP

Set of uA optimizations for VPEs to fully utilize L1-memory BW

1. **Decoupled VLSU interfaces**
2. **Dynamic Priority allocation & Shadow buffering**
3. **Improved vector chaining**
4. **L1 address scrambling**

<u>Goal:</u> Achieve at-the-roofline VPE performance for low Operational Intensity workloads

VPE architecture

1. Vector Register File

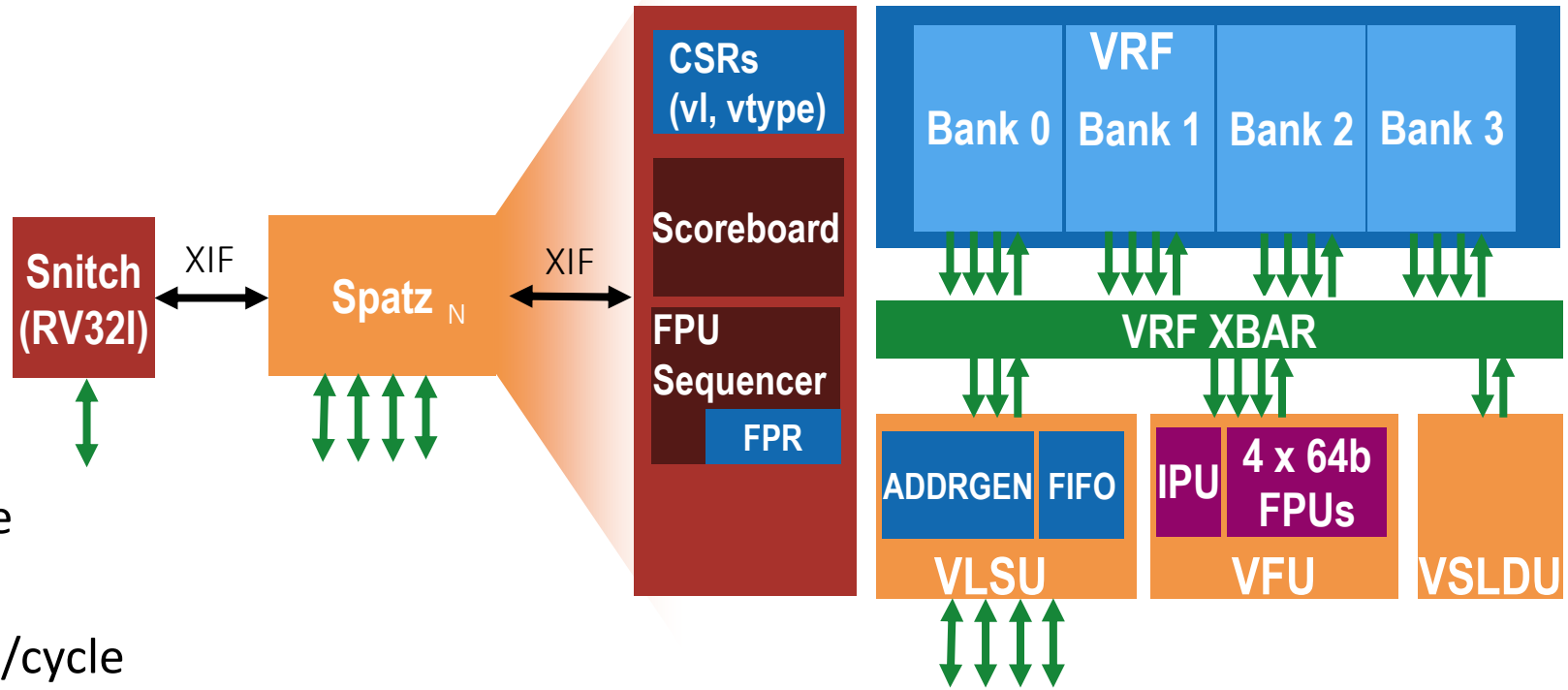
- Latch-based
- 256-bit R/W accesses
- 4 banks, 3R + 1W each

2. Execution Units

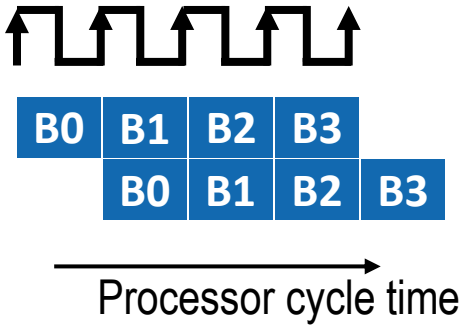
- FPUs + IPU (VFU)
  4 x 64-bit – 256b FMA/cycle
- Load-Store unit (VLSU)
  4 x 64b ports – 256b LD/ST /cycle

3. Controller

- Vector Chaining
- 1-bit credit based



vle64.v **v0**, (%0)
vfadd.vv **v2**, v1, v0

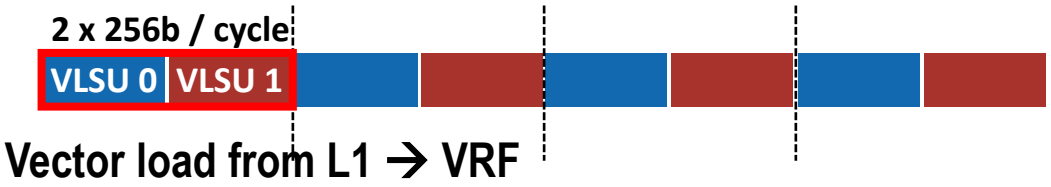


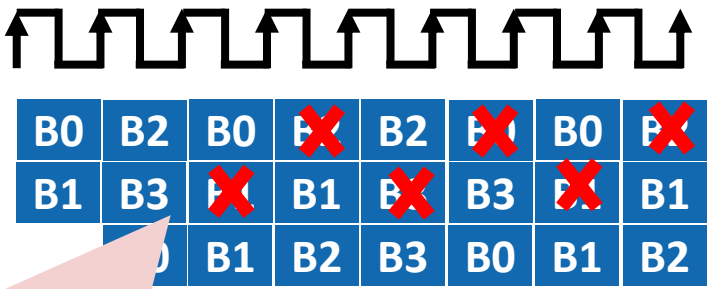
VRF Write v0 - VLSU
VRF Write v2 - VFU

Utilize multi-bank parallelism

Decoupled VLSU interfaces

What if BW to L1 is increased by 2x?



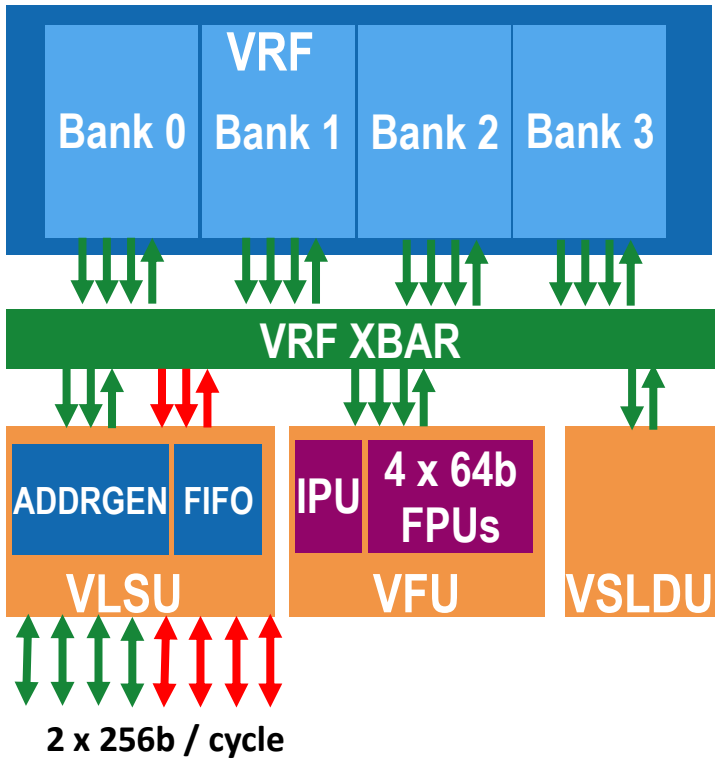


vle64.v **v0**, (%0)

vfadd.vv **v2**, v1, v0

VRF Write v0 – VLSU

VRF Write v2 – VFU



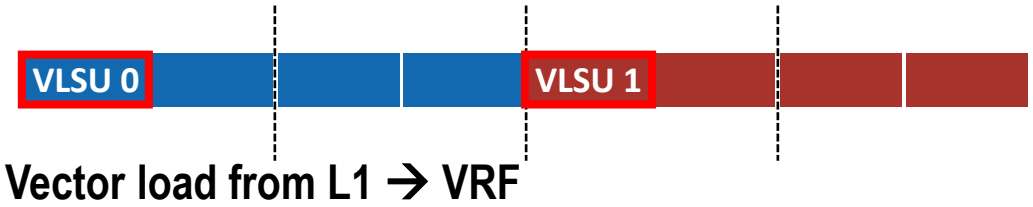
BW ≠ Compute ⟹ Structural Conflicts!

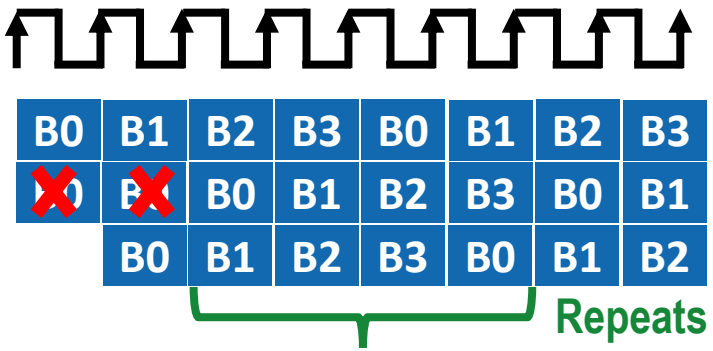
Only 50% memory bandwidth utilization

⟹ Need to eliminate structural conflicts!

Decoupled VLSU interfaces

***Solution:* Decouple the VLSU load interfaces**



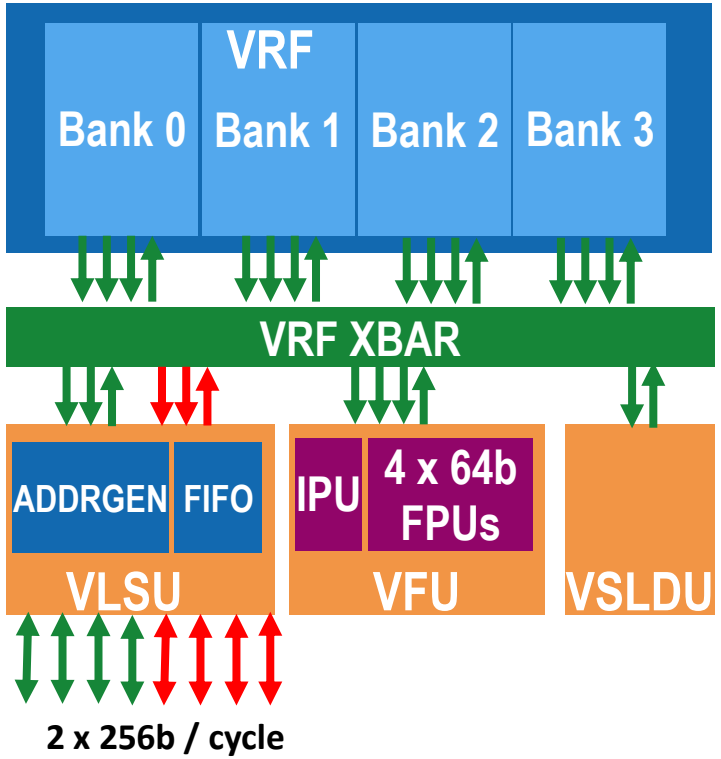


vle64.v **v0**, (%0)

vfadd.vv **v2**, v1, v0

VRF Write v0 – VLSU

VRF Write v2 – VFU

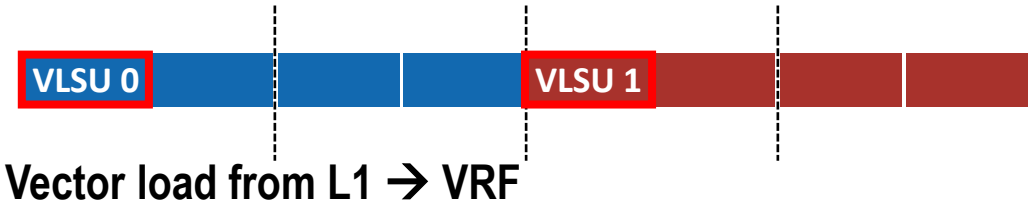


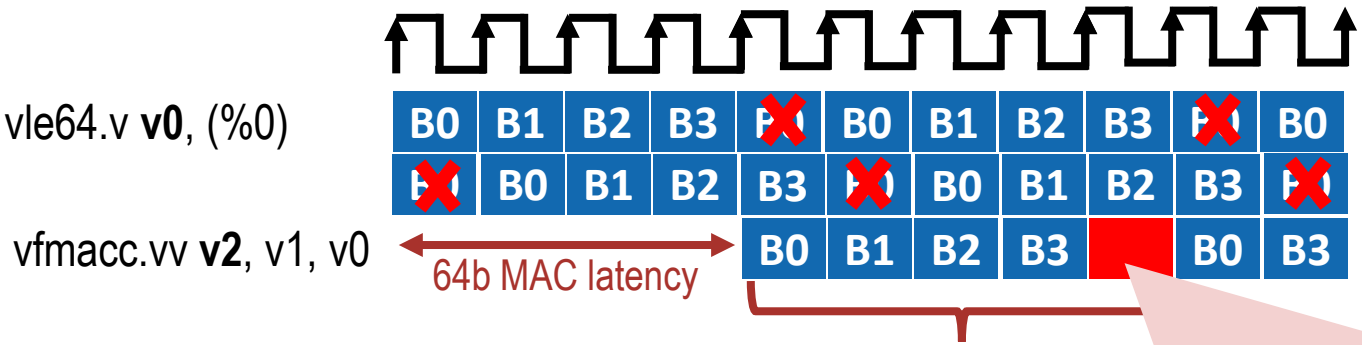
(1) Regularize VRF access strides

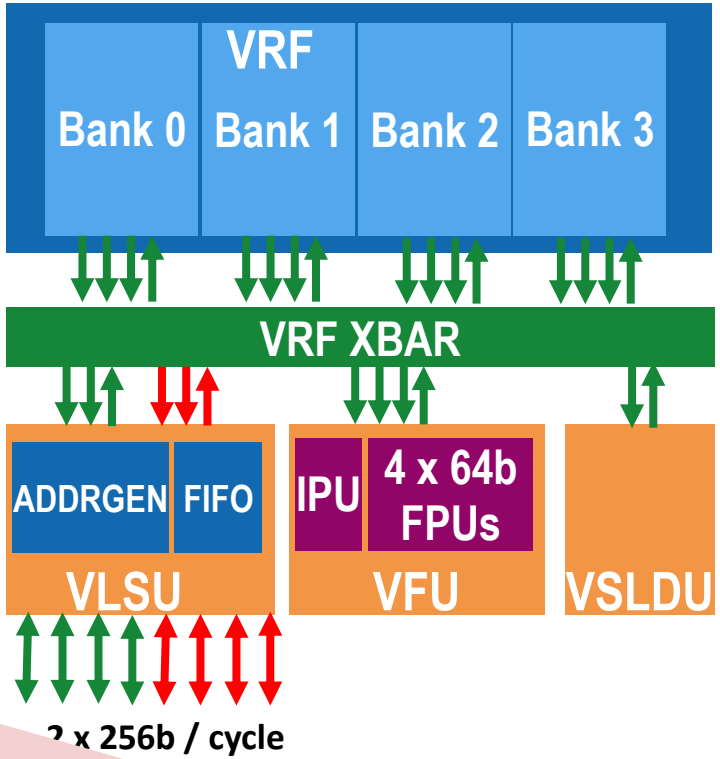
⟹ 100% Mem BW Utilization

Latency induced VRF conflicts

What about different datapath latencies?







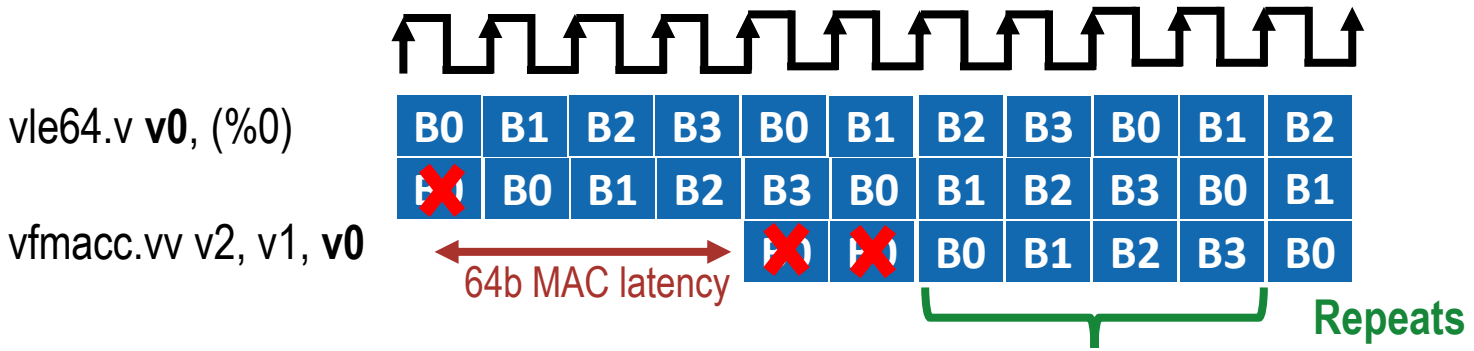
Conflicting VFU writes, block chained VFU reads,
Pipeline bubbles every 4 cycles (80% Peak Mem BW utilization)

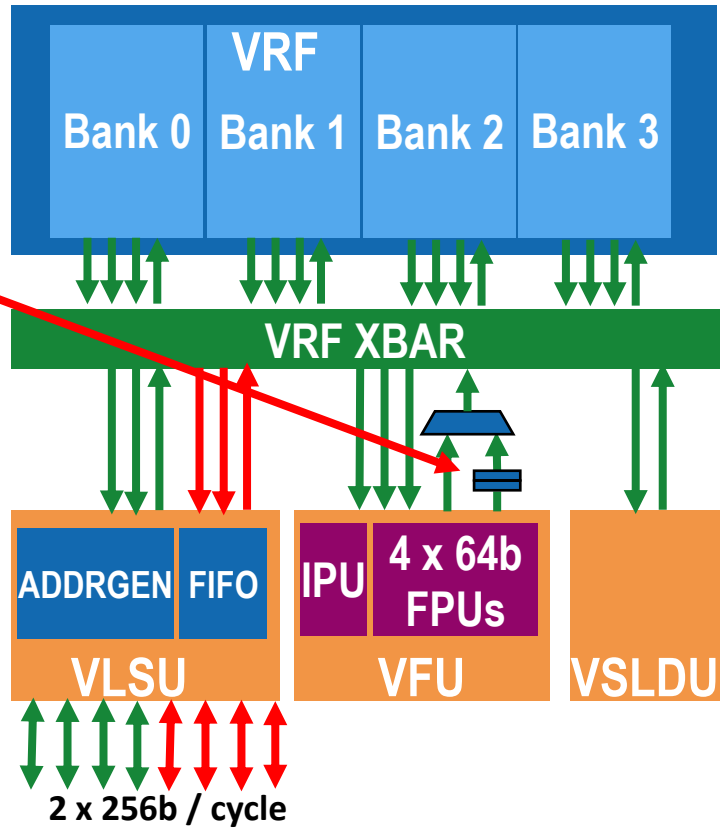
⟹ Need Datapath latency agnostic solutions!

Dynamic Priority allocation & Shadow buffers

Solution: ***Need buffer to tolerate conflicts!***

1. ***Buffer VFU response during conflicts***
   Avoid stalling VFU!
2. ***Prioritize VFU if buffer is full***





Buffering allows full throughput VFU operation,
even during pending write-back (more OOO capability in vector unit)

(2) Dynamic Priority allocation & Buffering
⟹ 100% Mem BW Utilization

$\text{Spatz}_{2\times\text{BW}-\text{TROOP}}$ Performance

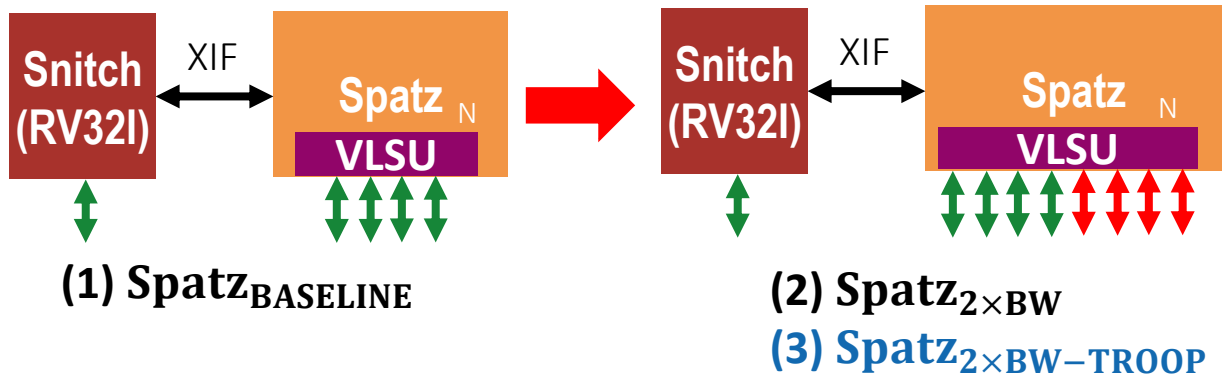
Evaluate common micro-kernels

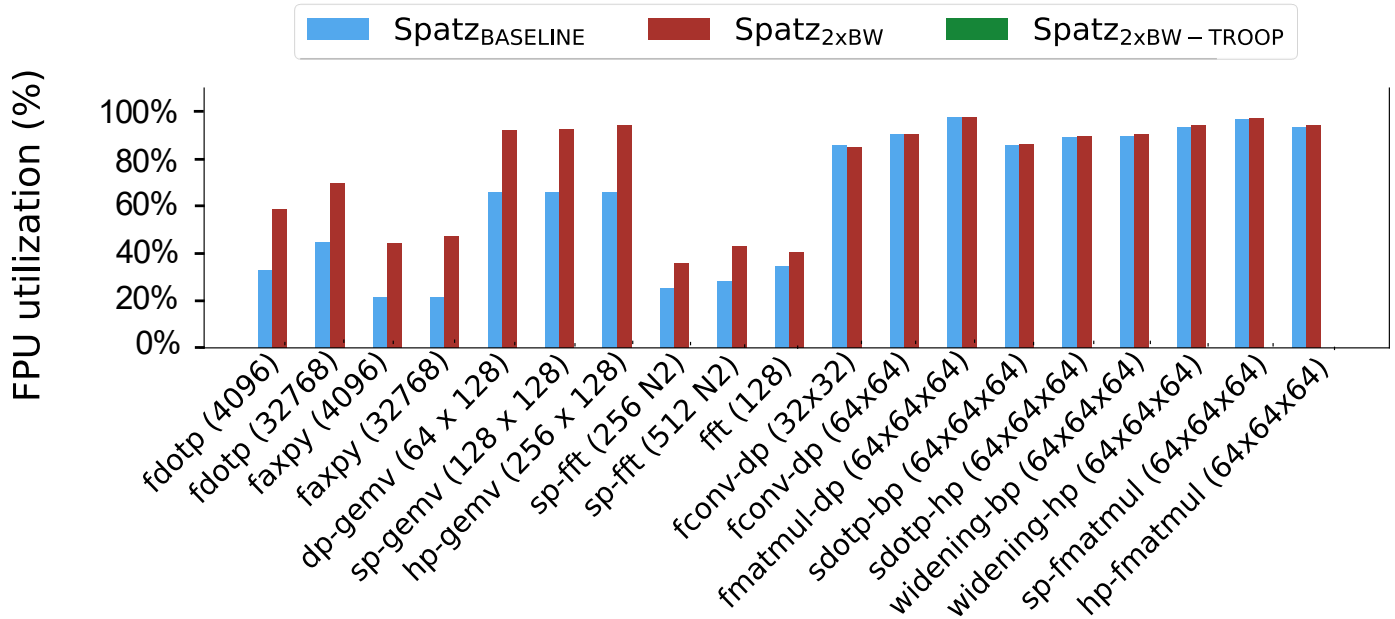
Memory-intensive : DOTP, AXPY, GEMV, FFT

Compute-intensive : MMUL, CONV

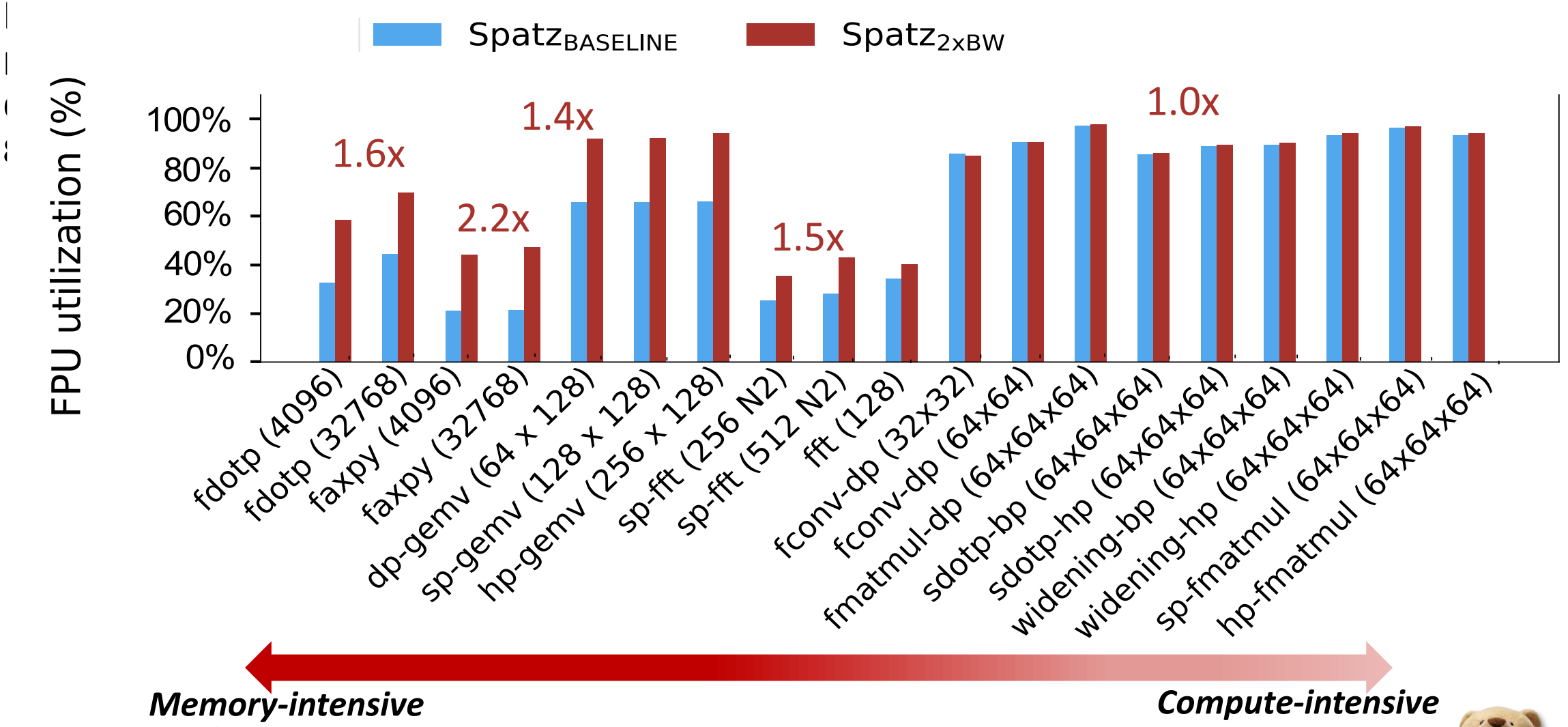
→ 8/16/32/64b precision,

→ Varying Problem sizes

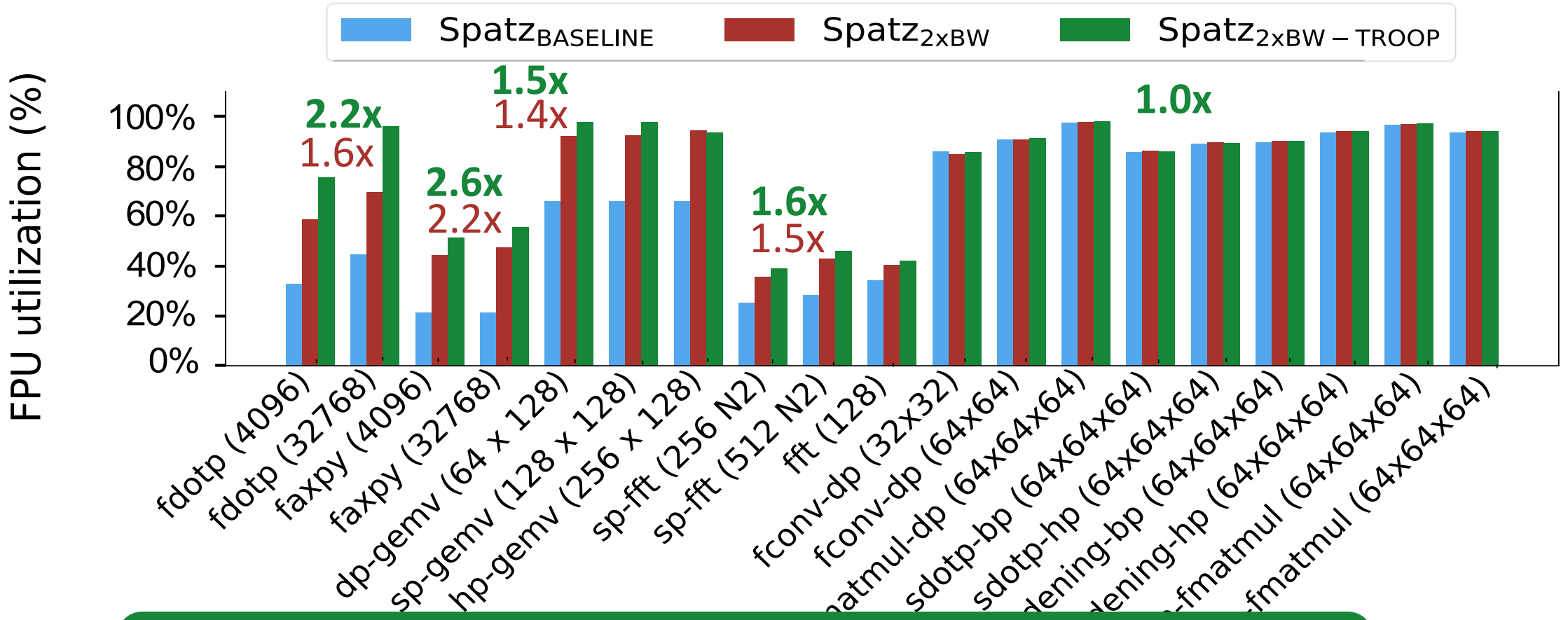




$Spatz_{2\times BW-TROOP}$ Performance



$Spatz_{2\times BW-TROOP}$ Performance

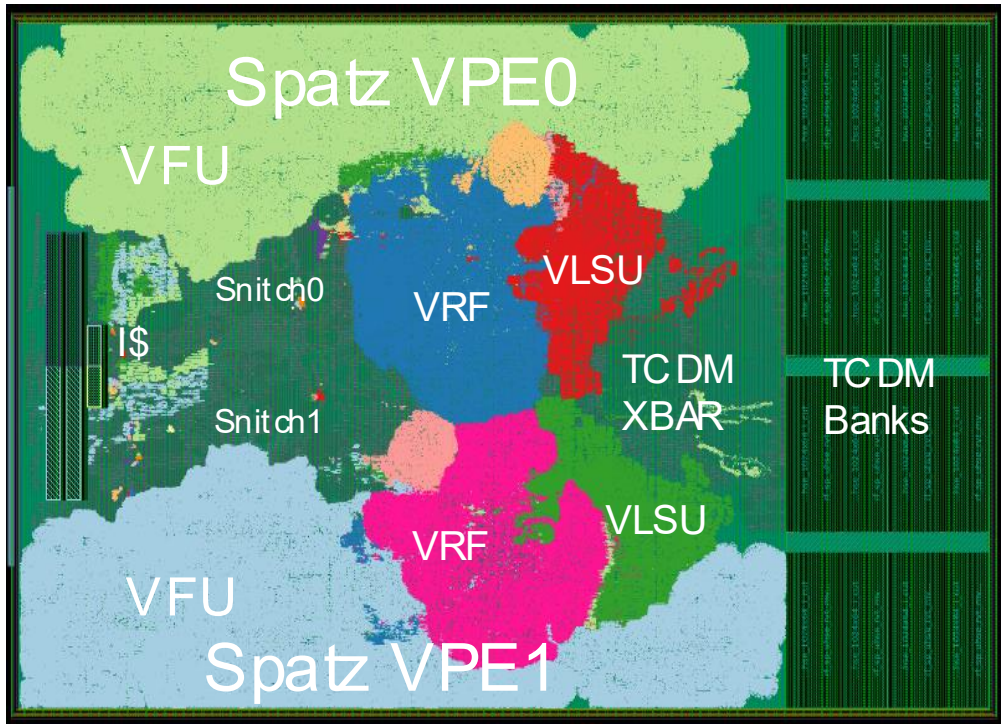


TROOP necessary to fully-exploit higher L1-memory BW!

Memory-intensive *Compute-intensive*

$\text{Spatz}_{2\times\text{BW}-\text{TROOP}}$ Physical Implementation

- **$\text{Spatz}_{\text{BASELINE}}$ & $\text{Spatz}_{2\times\text{BW}-\text{TROOP}}$ (2-core configuration, 128 KiB shared-L1)**
- **GF12 nm, Synopsys Fusion Compiler, 1 GHz target (SS/0.72V/125C)**



Freq: 1.4 GHz (TT/0.8V/25C)
No Added critical paths

Area: 4.4 MGE (<7% increase w.r.t $\text{Spatz}_{\text{BASELINE}}$)
Major contribution - 2xBW on VLSU and TCDM XBAR

Energy Efficiency:
Up to 45% improvement w.r.t $\text{Spatz}_{\text{BASELINE}}$
26 → 38 DP-GFLOPS/W DOTP (4096 x 64b)

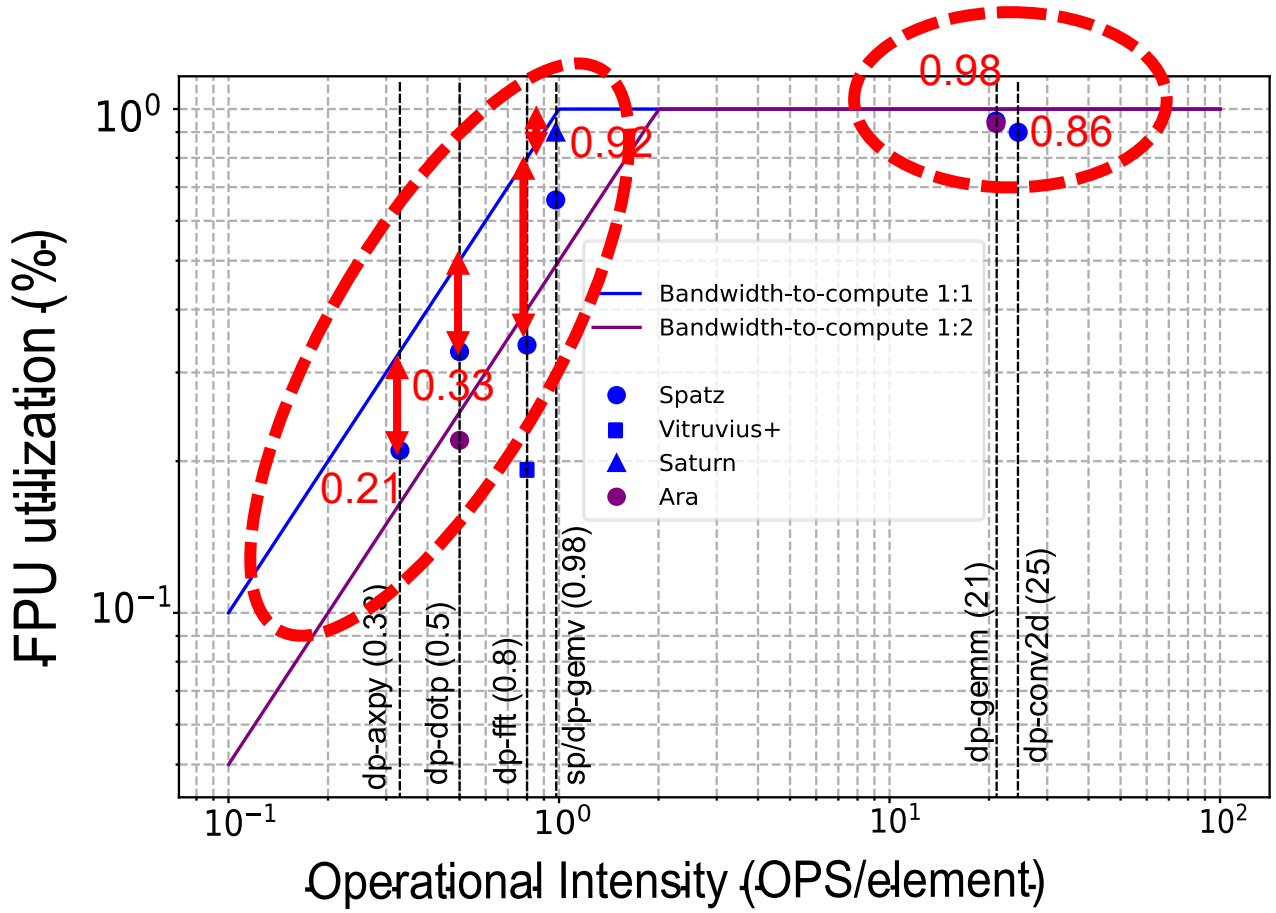
Comparison with SoA

Challenge:

✔ **SoA VPEs optimized for compute-intensive workloads**

✘ **Achieve poor FPU utilizations far from roofline for memory intensive workloads**

VPE	BW:COMP	uA Features
Saturn	1:1	Dynamic Instr. Scheduling
$\text{Spatz}_{\text{BASELINE}}$	1:1	Streamlined small-VLEN
Vitruvius+	1:1	OOO vector chaining
Ara2	1:2	Long-VLEN, scalability
NEC VE30	1:6	HPC, Large VRF, wide OOO

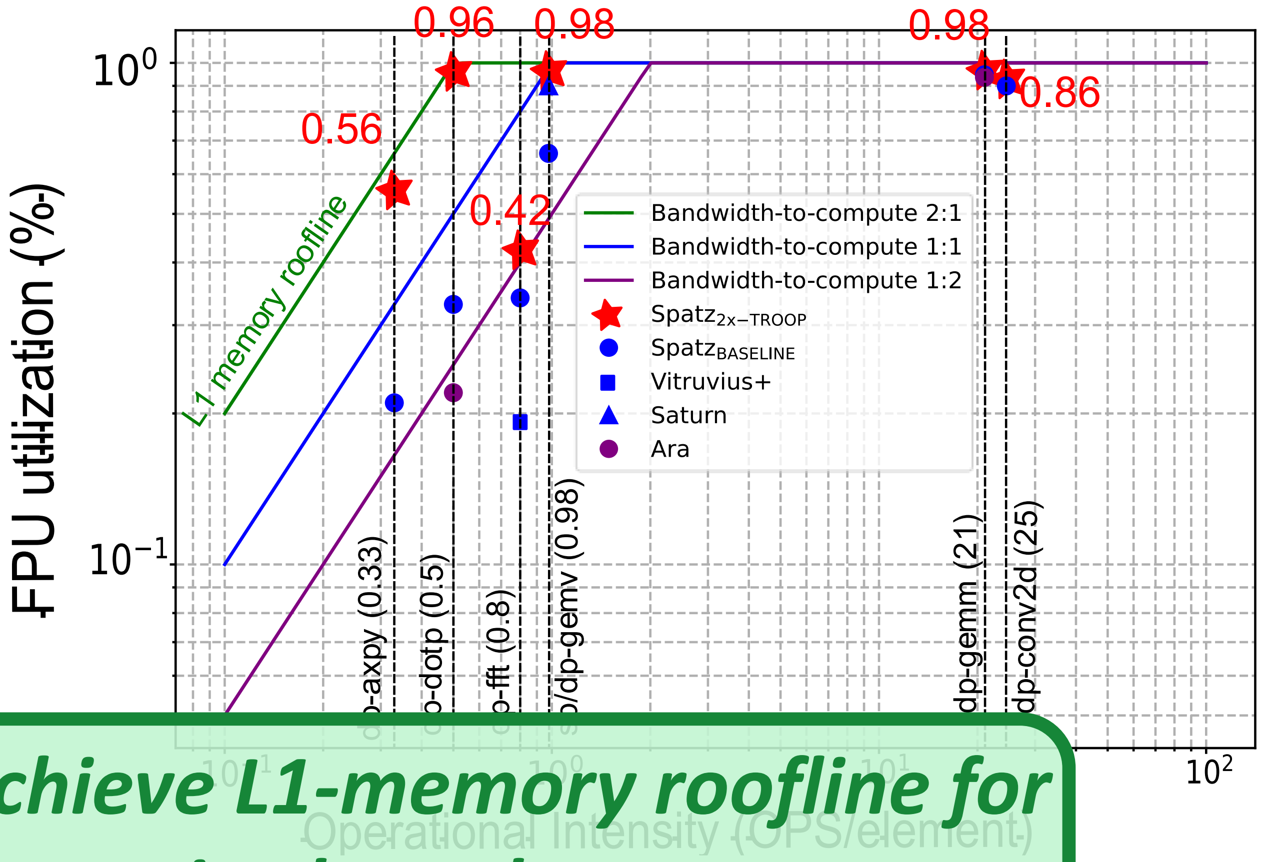


Comparison with SoA

Challenge:

- ✔ **SoA VPEs optimized for compute-intensive workloads**
- ✔ **Achieve at-the-roofline FPU utilizations for memory intensive workloads**

VPE	BW:COMP	uA Features
Saturn	1:1	Dynamic Instr. Scheduling
$Spatz_{BASELINE}$	1:1	Streamlined small-VLEN
Vitruvius+	1:1	OOO vector chaining
Ara2	1:2	Long VLEN, scalability
NEC VE30	1:6	HPC, Large VRF, wide OOO
$Spatz_{2x-TROOP}$	2:1	Decoupled VLSU, Dynamic priority and shadow buffering



With TROOP VPEs can achieve L1-memory roofline for memory-intensive kernels

We present TROOP…

- **Set of uA optimizations targeting today's memory-intensive workloads**
- **We overcome structural limitations in SoA VPEs and fully leverage L1-memory bandwidth**
- **On the compact SoA Spatz VPE,**
  Performance: 2.6x AXPY, 2.2x DOTP, 1.5x GEMV, reaching at-the-roofline Performance!
  Low-cost: (<7% area inc.), 1.4 GHz (TT/0.8V/25C)
  High Energy Eff.: DOTP – 38 DP-GLOPS/W (+45%), GEMV – 52 DP-GFLOPS/W (+9%)

Open source : https://github.com/pulp-platform/spatz/

ETH zürich PULP Parallel Ultra Low Power ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA SwissChips

Navaneeth Kunhi Purayil nkunhi@iis.ee.ethz.ch
Diyou Shen dishen@iis.ee.ethz.ch
Dr. Matteo Perotti mperotti@iis.ee.ethz.ch
Prof. Dr. Luca Benini lbenini@iis.ee.ethz.ch

Institut für Integrierte Systeme – ETH Zürich
Gloriastrasse 35
Zürich, Switzerland

DEI – Università di Bologna
Viale del Risorgimento 2
Bologna, Italy

ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

pulp-platform.org
@pulp_platform
company/pulp-platform
youtube.com/pulp_platform