PULP 10-Year Anniversary

Lugano, June 5-6, 2023

Boosting PULP with Vector and Transprecision Capabilities

Matteo Perotti
Luca Bertaccini

IIS, ETH Zurich, Switzerland,

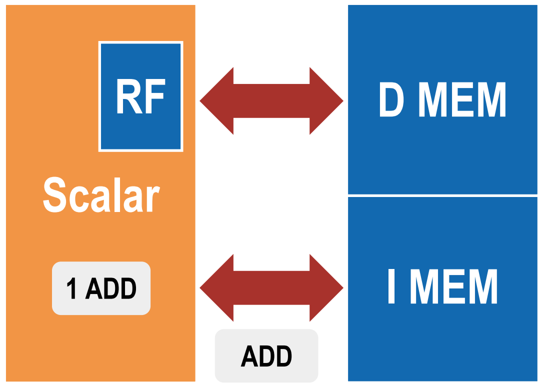
PULP Platform
Open Source Hardware, the way it should be!

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

SIMD processor, a friendlier example

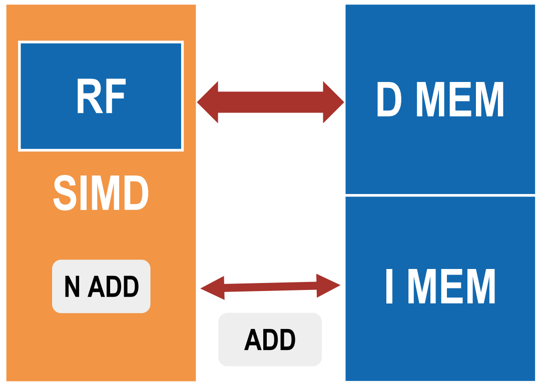
SCALAR CORE

- One instruction – One Operation
  - BW and Power on the I-MEM
- RF size is usually fixed
  - One data element per entry
  - Too small for highly-intensive WL
    - BW and Power on the D-MEM



SIMD CORE

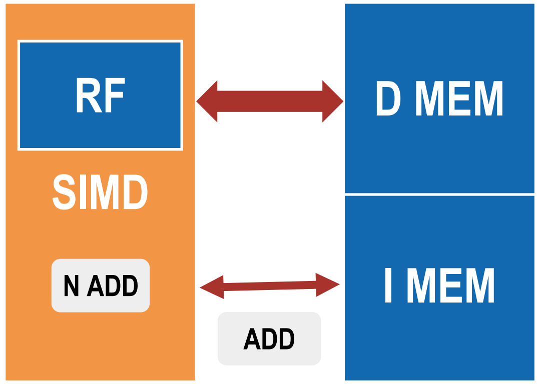
- One instruction – Multiple operations
  - Lower BW and Power on the I-MEM
- RF size is larger
  - Multiple data elements per entry
  - Better buffering, exploit locality
    - Lower BW and Power on the D-MEM



Vector Processor – A flexible and efficient choice

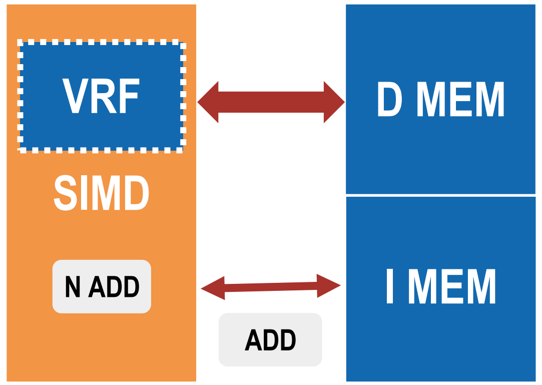
SIMD CORE

- Vector Length (VL) encoded in the instruction!
  - Support new VL? Extend the ISA!
    - e.g., add128, add256, ...
  - RF is not flexible



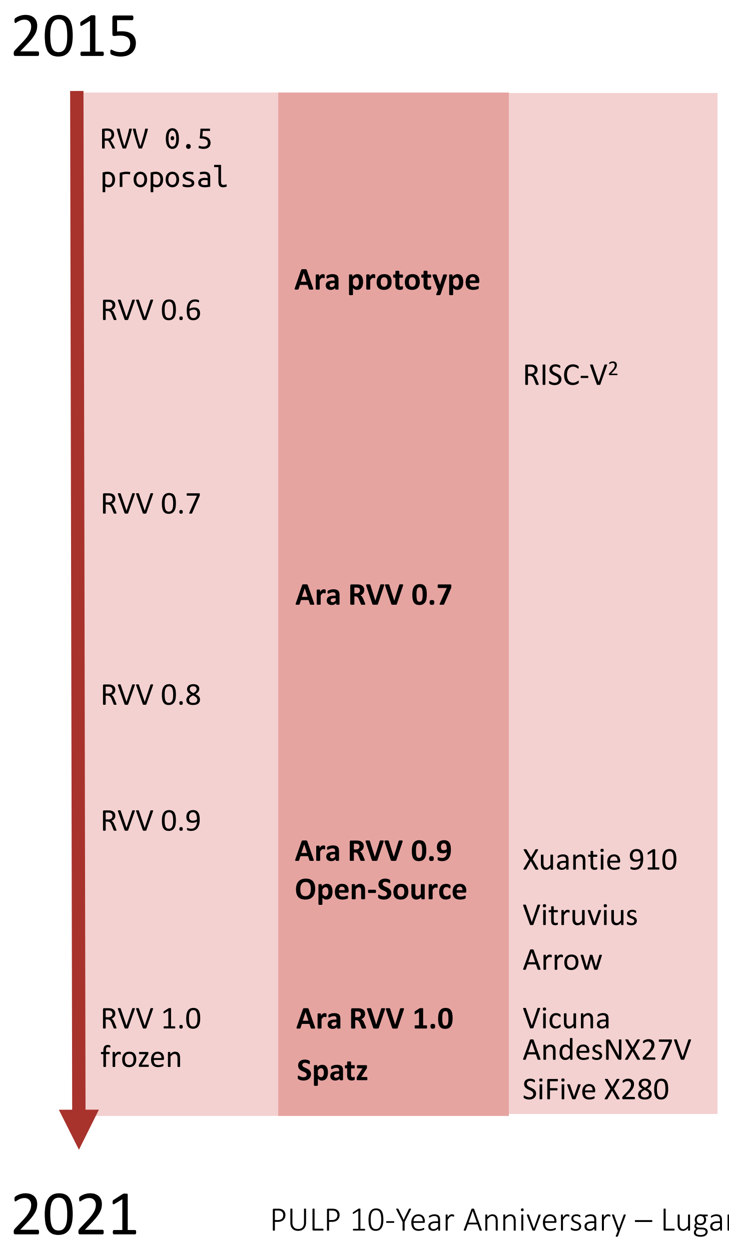
VECTOR CORE

- Programmable VL
  - Flexible Vector RF (VRF)
  - VRF size decoupled from DP width



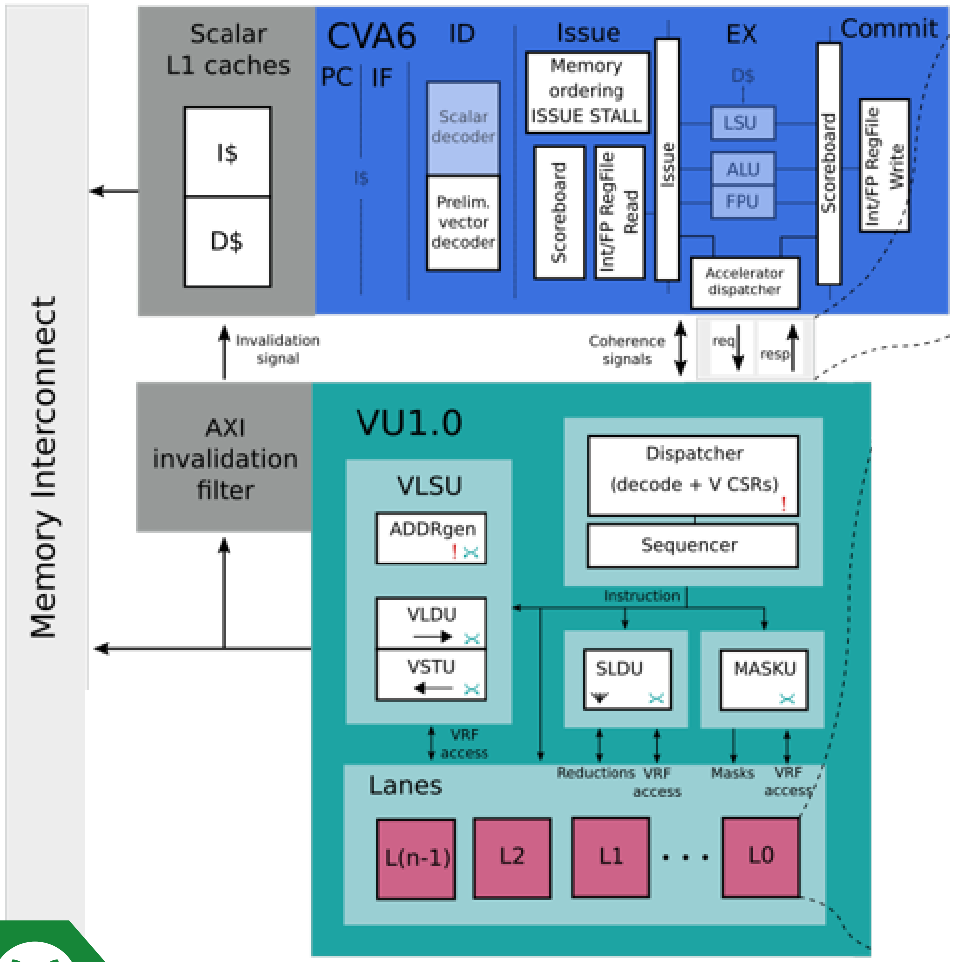
RISC-V RVV ISA

- Early draft in 2015
- Years of refinment. Now it's frozen.
- **Known programming model**
- Add **new VRF** with **parametrizable size**
- **Huge** and comprehensive ISA (+300 instructions)
  - VRF setup instructions (VL, element width, ...)
  - Arithmetic instructions (int, fixpt, fp)
  - Memory operations (unit-stride, strided, indexed)
  - Predicated execution (masks)
  - Permutation instructions (slide, reorder) **Vector length agnostic** – High reusability!



Ara - PULP Application-Class Vector Processor

- **CVA6 + Ara**: RV64GCV
- **CVA6**
  - Access to I-MEM
  - Non-speculative V Dispatch
- **Ara**
  - Private VLSU (vload + vstore)
  - Slide Unit (permutations)
  - Mask Unit (predication)
  - Lanes (computation + VRF)
- OS Support soon…



github.com/pulp-platform/ara

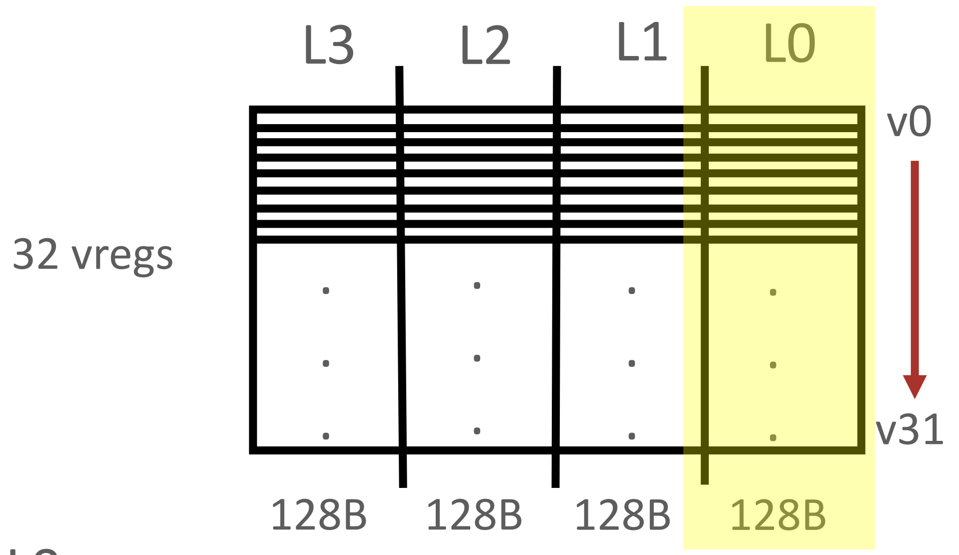
Ara’s heart - The Lane

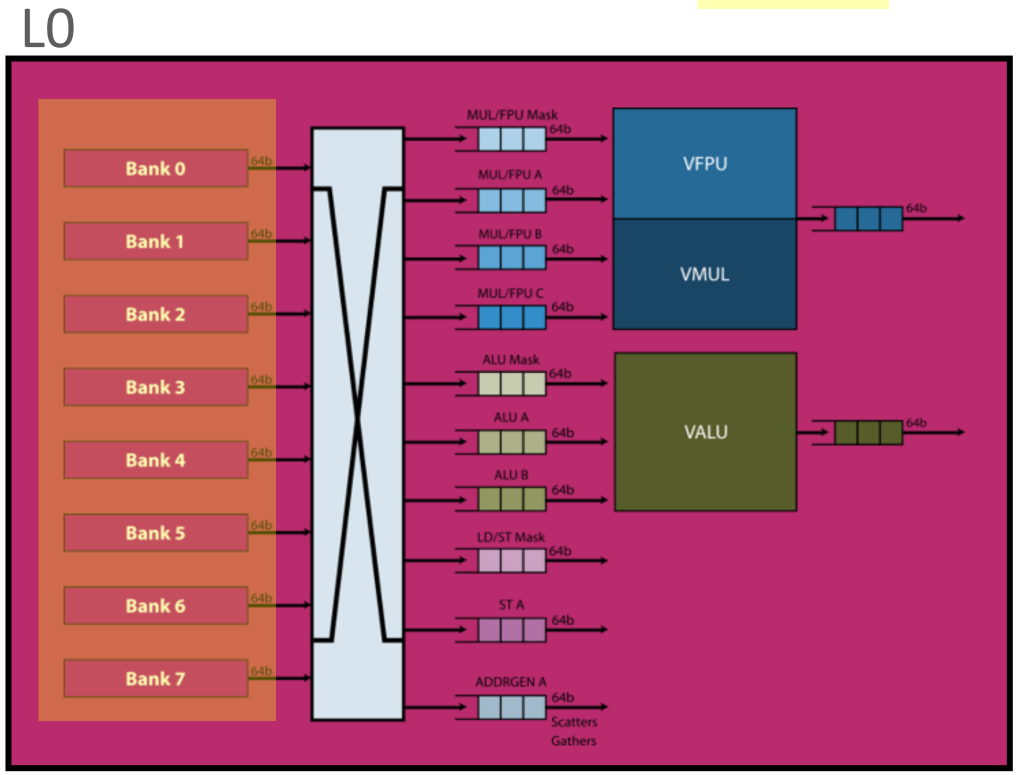
Ara default **VRF**: 4 KiB * #Lanes

- 32 Vector Register
- Vector Register can be merged
- Split among the lanes

Each Lane:

- VRF chunk (8 1R/W SRAM Banks)
- VALU, SIMD MUL, FPU
- Maximize locality
  - Element 0 ramains in L0

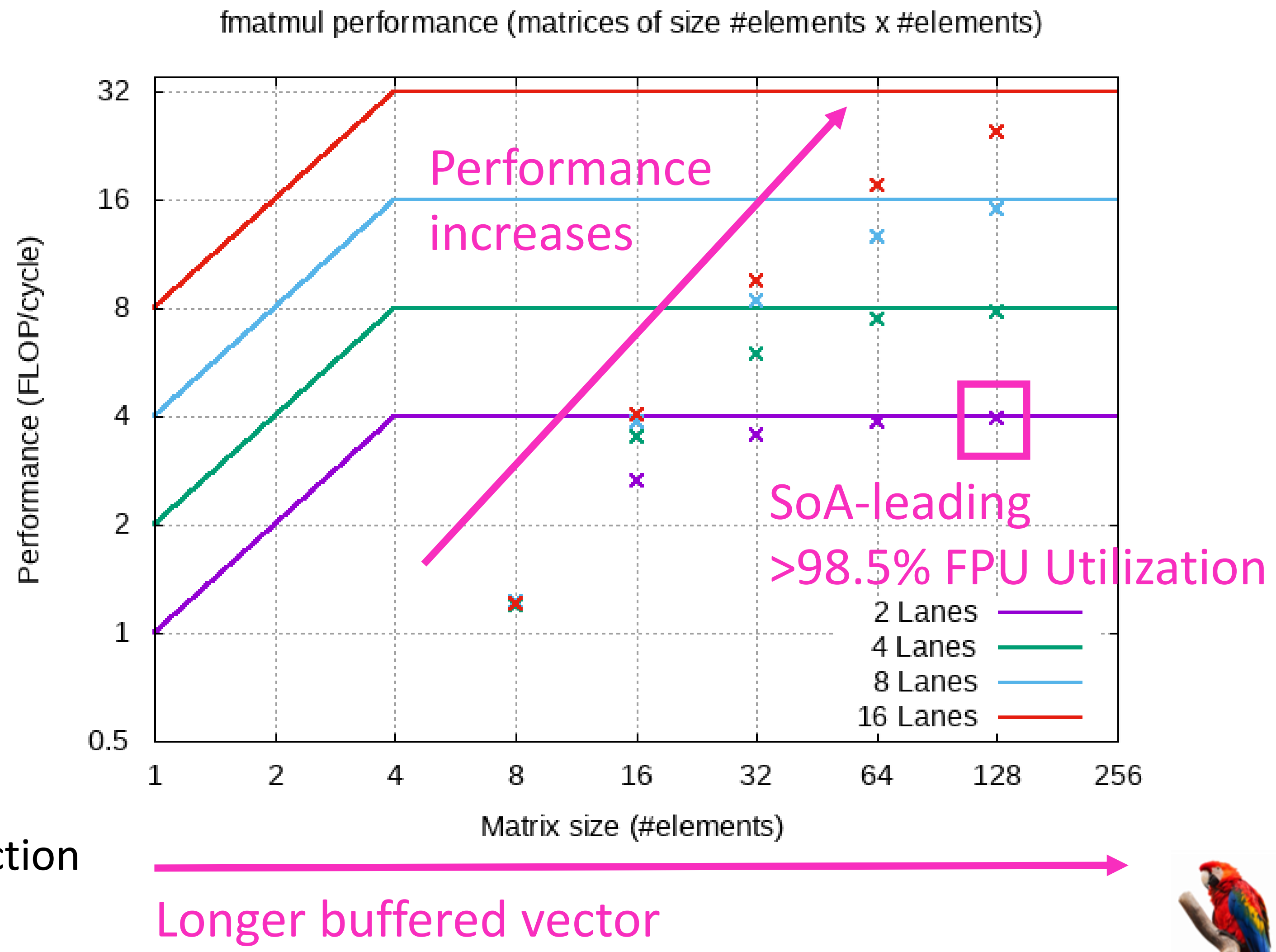




ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

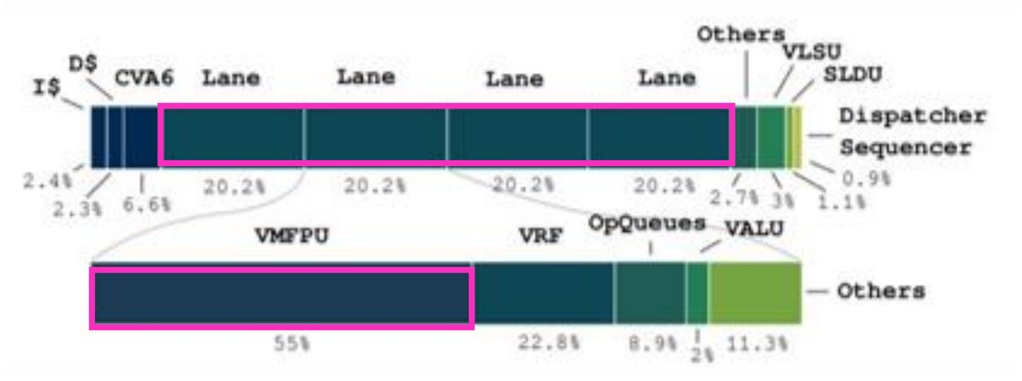
Ara on the Benchmark

- Up to **32 DP-FLOP/cycle**
  - With 16 Lanes
- FP-matmul
  - A[N][N] * B[N][N]
  - X-axis: Matrix Size N
  - Always computation-bound
- Longer vector, better performance
  - Amortize setup time for V instruction
  - Amortize CVA6 + $ non-idealities

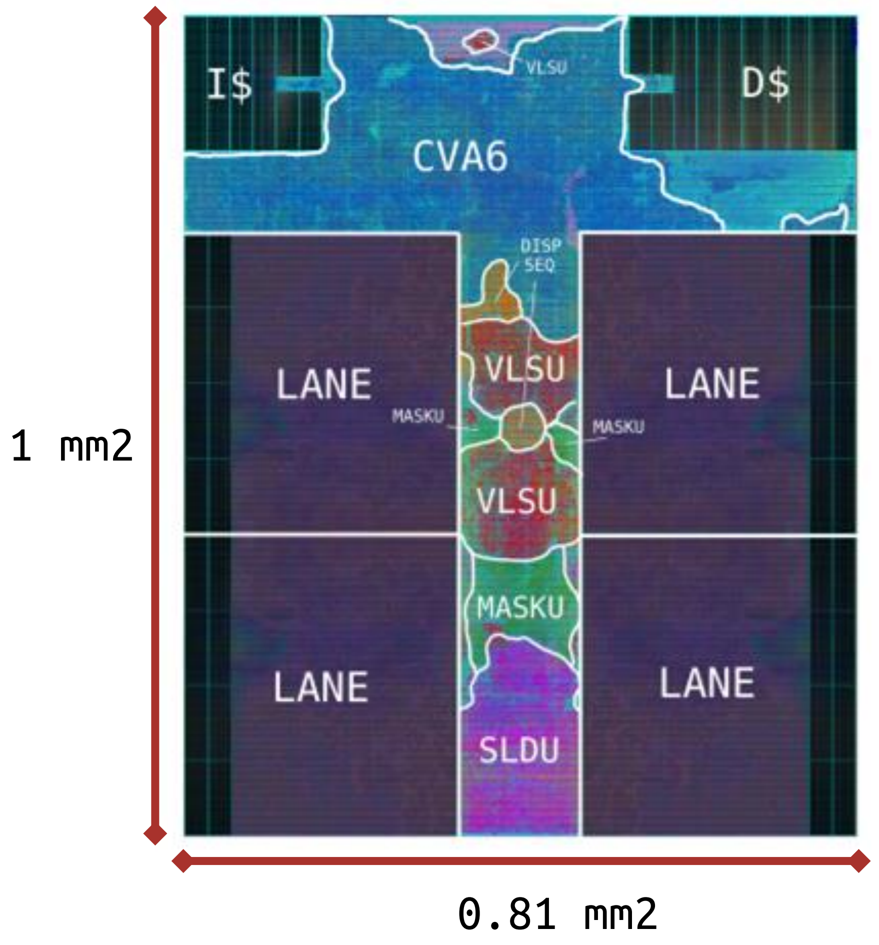


The physical Ara – Fast and Efficient

- GF22 FDX FD-SOI (0.8V, 25*C)
- SS frequency: 950 MHz
- TT frequency: 1.34 GHz
- Power: 280 mW
    - 1.34GHz, during fp-matmul
- Efficiency: 37.6 DP-GFLOPS/W

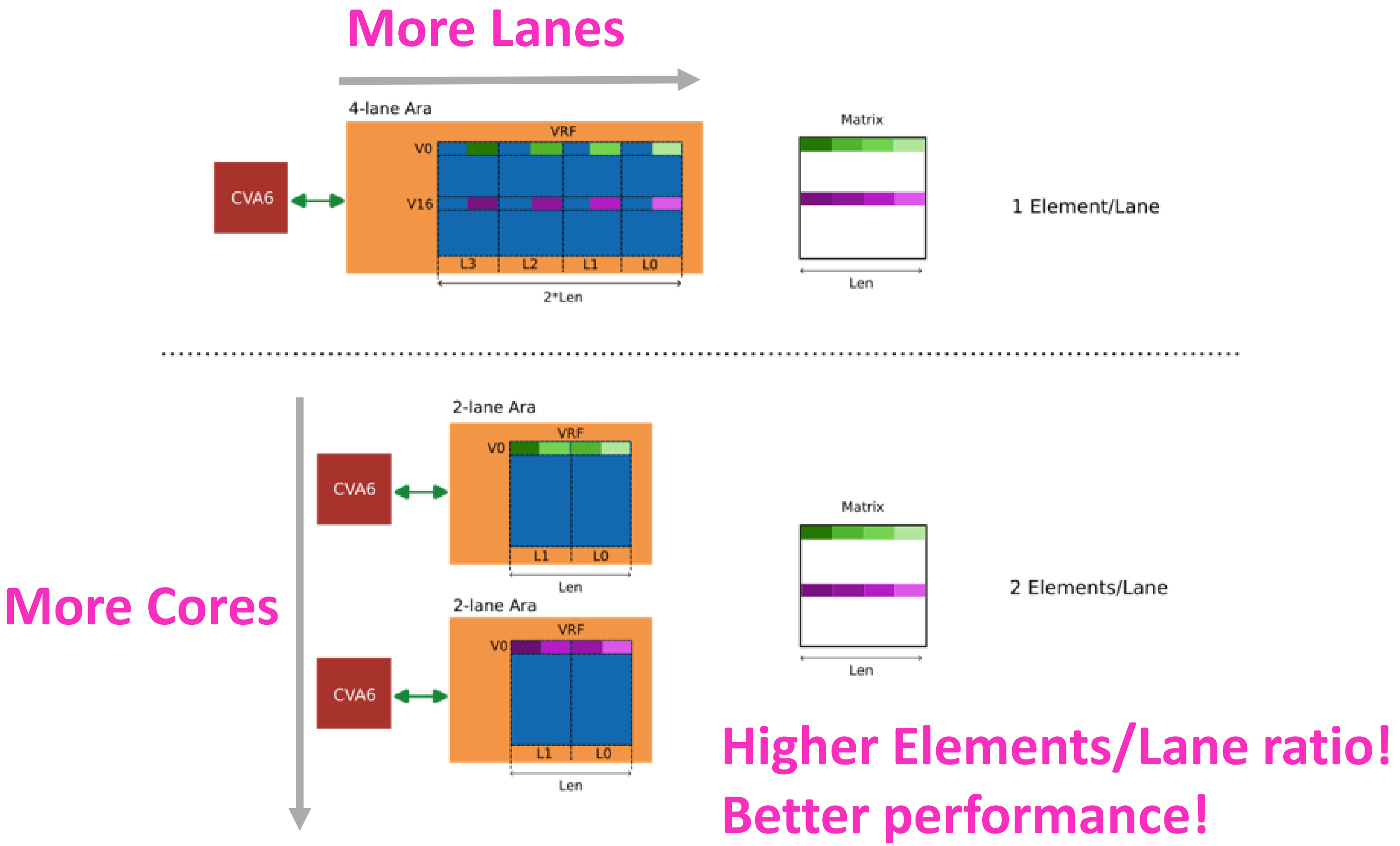


Ara 4 Lanes



Scaling up – More Lanes or more Aras?

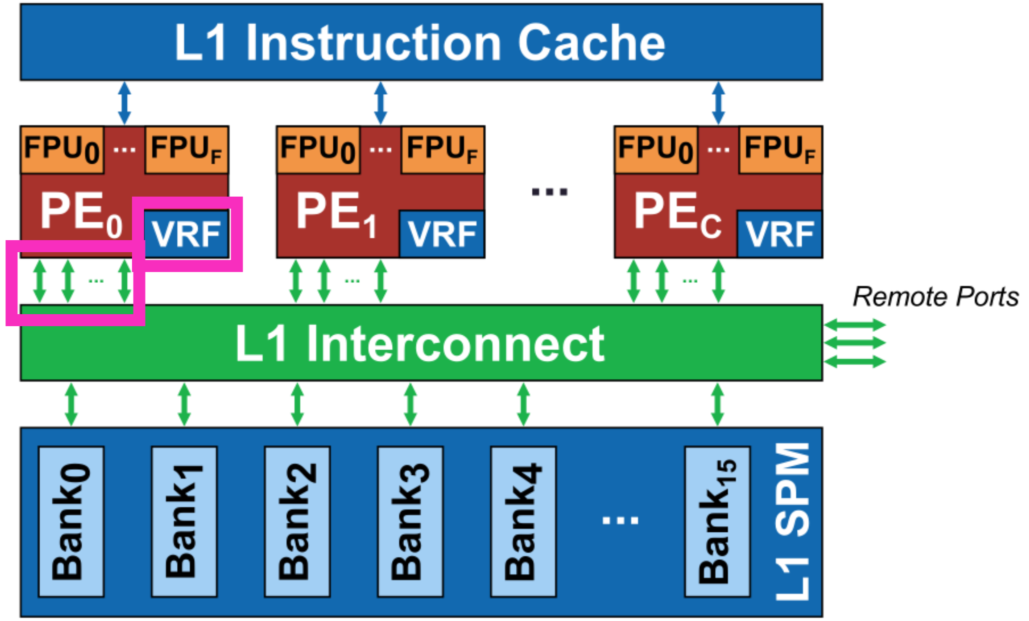
- **More lanes in one Ara**
  - Maximum #Lane/#CVA6 ratio
  - CVA6 issue-rate limitation
  - No synch overhead
  - Minimum memory traffic
  - Need longer vectors to keep high Element/Lane ratio
  - Hard to scale more than 8 lanes
- **More Aras, fewer lanes/Ara**
  - Worse #Lane/#CVA6 ratio
  - Lower issue-rate limitation
  - Synch overhead
  - Increased memory traffic
  - Easier to keep high Element/Lane ratios even with shorter vectors



Higher Elements/Lane ratio!
Better performance!

An embedded vector unit: Spatz

- Tiny embedded vector machine to focus on the DLP
- **Spatz$_F$** has **F** FPUs on board
- **Spatz$_F$** replaces **F** Snitch+SSR cores
- Spatz's **VRF** is an **L0 memory** inside the cluster
  - More buffering means **RELAXED BW** on the **L1** interconnect! (Kung's Balance between L0 size and L1 BW)
- Easier routing of the interconnect
- Standalone cluster + Mempool integration!



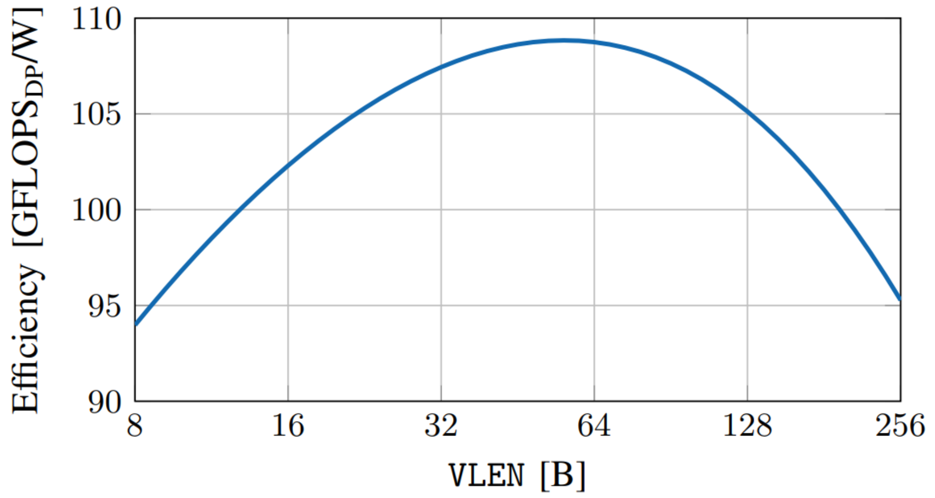
L1-BW vs. VRF-size trade off!

Snitches + SSRs: **L1 BW**

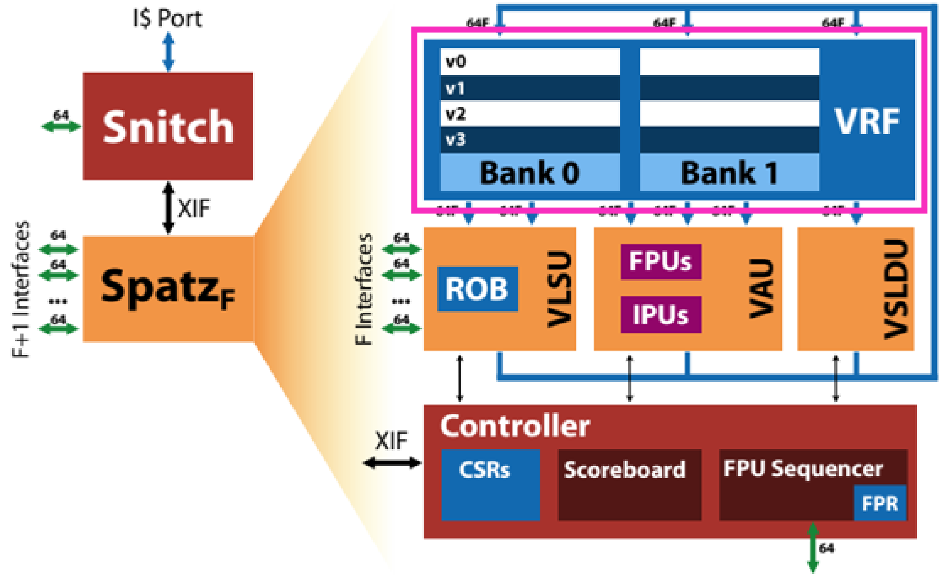
Spatz: **VRF size**

Spatz Architecture

- Subset of Zve64d (**RVV for embedded**)
- **Centralized Latch**-Based SCM Vector Register File (VRF)
  - 2 Banks/VRF - 2 KiB/Bank
  - 32 vector registers, 64 B each
  - Optimized for Energy Efficiency
- F Memory interfaces (half of F Snitches' BW)
- Support for:
  - 64-bit floating-point + Mixed Precision
  - FP-DOTP operations (more on this in a while...)
  - Reductions
  - Indexed memory operations

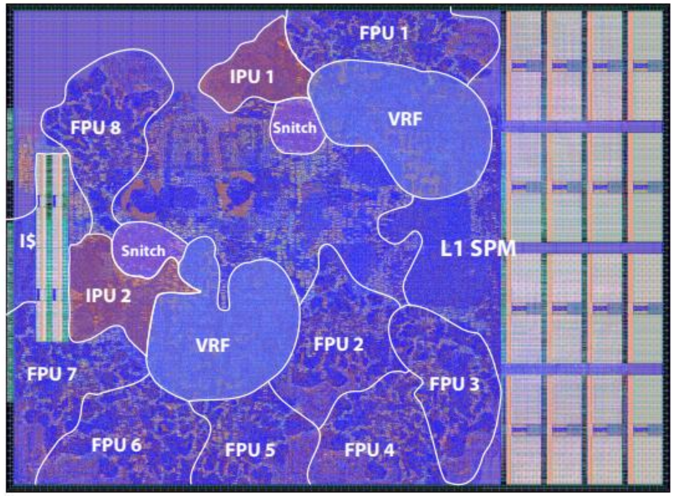


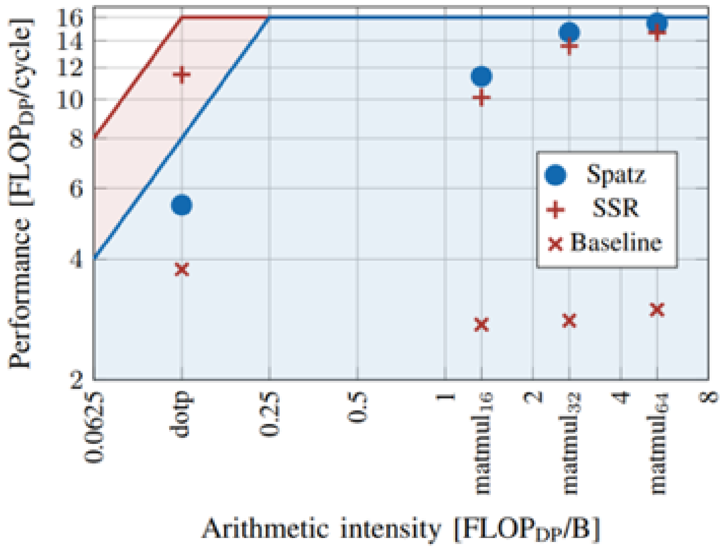
VLEN = 64 B - Max Efficiency!

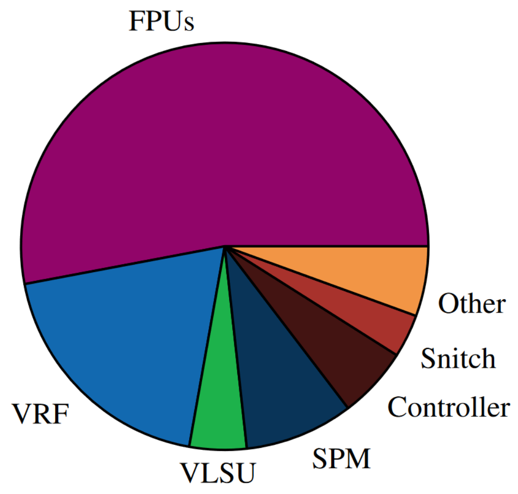


Spatz: highly-efficient shared-L1 cluster

- Configuration with 8 64-bit multi-precision FPUs and 128 KiB of L1
- **Performance:**
  - **15.6 GFLOPS** @ 1 GHz in GlobalFoundries' 12LPP process for a matrix multiplication kernel
- **Efficiency:**
  - **FPUs** consume 97 mW, 54% of the total power
  - **88 GFLOPS/W** at typical operating conditions







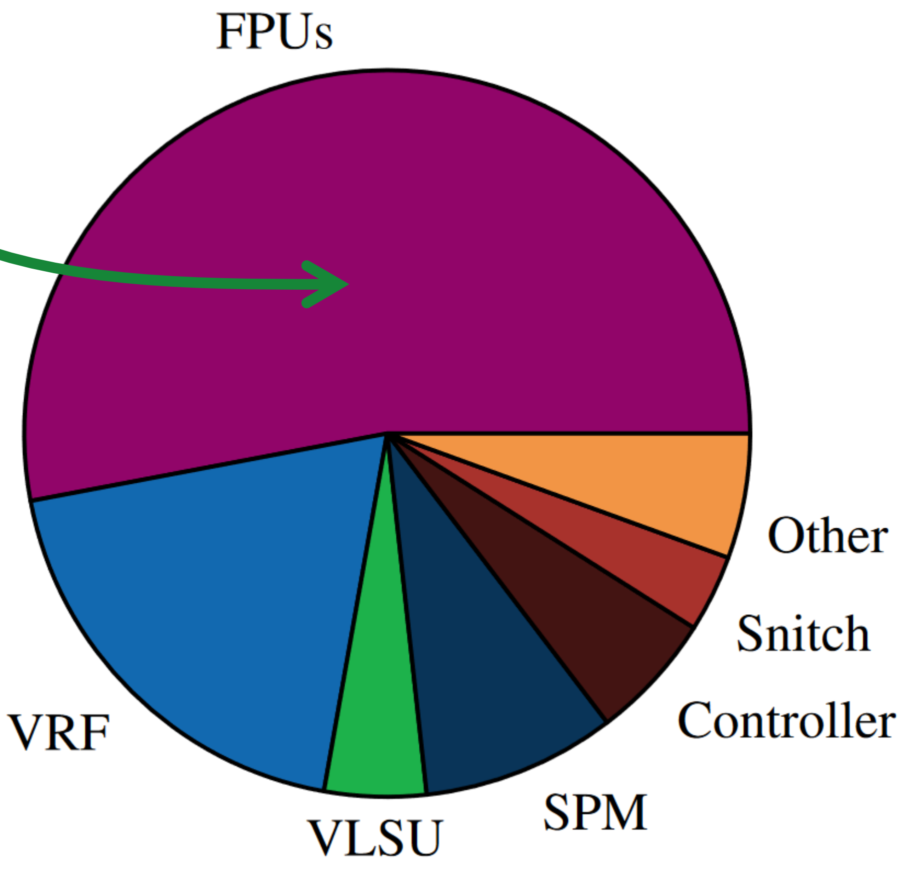
Can We Do More to Increase our Energy Efficiency?

- Already **more than 90% FPU utilization** and **more than 50% power** dedicated to the FPU

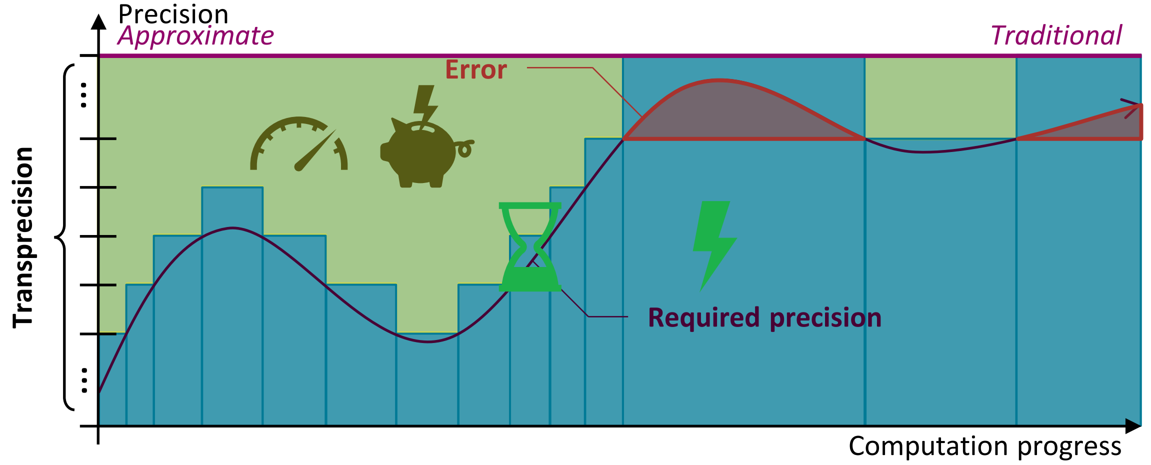
- Not much room for a higher FPU utilization, neither for further decreasing the power dedicated to control
- BUT we can increase the FPU efficiency by precision tuning

Spatz Power Breakdown

(12nm tech, 1 GHz, TT, 0.8V, 25°C)



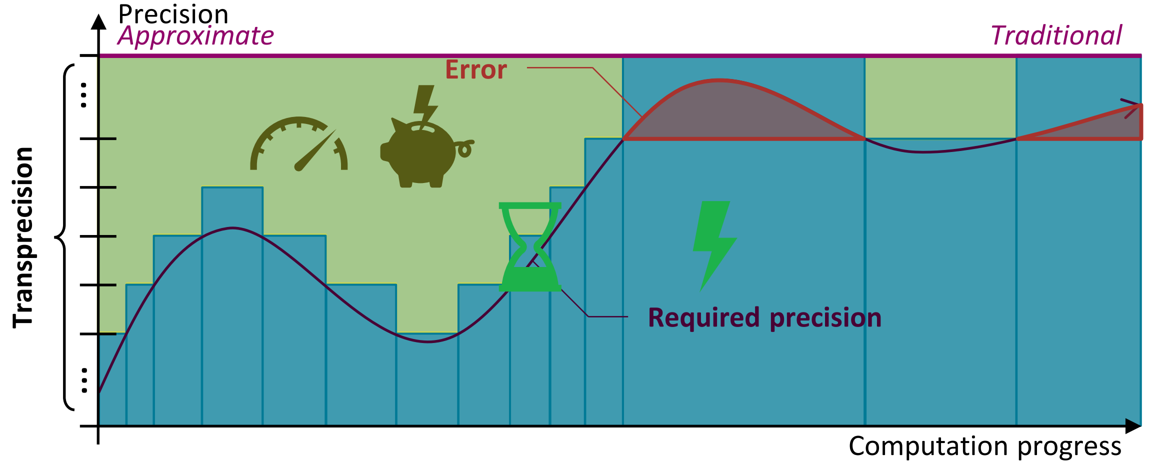
Transprecision Computing: More than just Approximation



S. Mach, "Floating-Point Architectures for Energy-Efficient Transprecision Computing"

- **Conservative** with precision (Use largest precision **everywhere** & **always**)
- Approximate Computing?
- **Transprecision** Computing:
  - Just right precision **anywhere** & **anytime**
  - Potential energy savings & speedup

Transprecision Computing: More than just Approximation



S. Mach, "Floating-Point Architectures for Energy-Efficient Transprecision Computing"

- **Conservative** with precision (Use largest precision **everywhere** & **always**)
- Approximate Computing?
- **Transprecision** Computing:
  - Just right precision **anywhere** & **anytime**

eedup

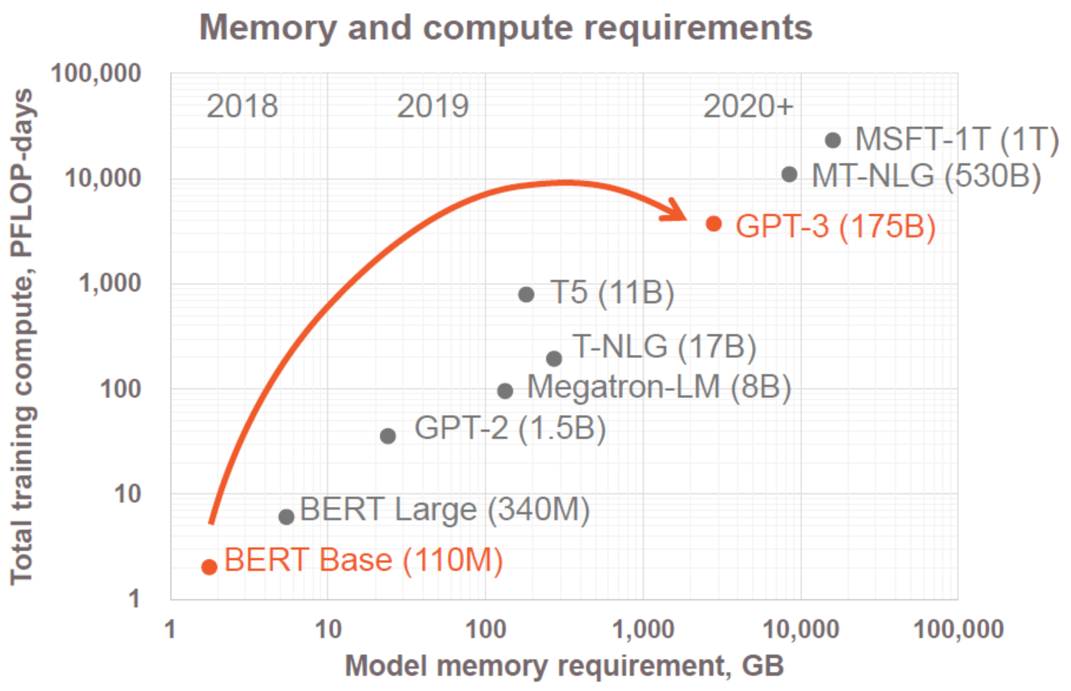
Need for multi-format support and mixed-precision instructions

Neural Network Training Pushing for Low, Mixed-Precision

- NN models' memory and compute requirements are rapidly growing

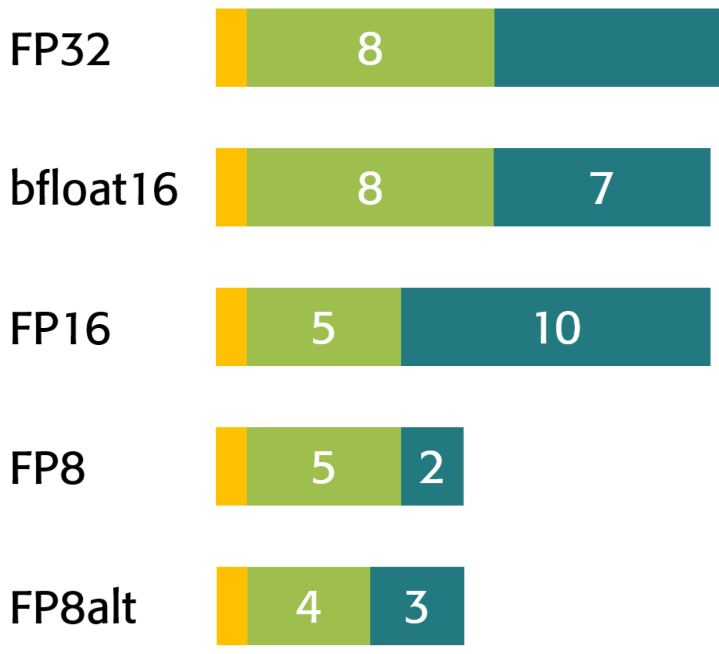
To efficiently compute these workloads, need for advancements in:

- Process technology
- Architecture (chiplet, tensor cores, HBM)
- Microarchitecture (sparsity, **low-precision**)
- Cluster scale-out



S. Lie, "Thinking outside the die: Architecting the ML accelerator of the future"

Low-Precision Floating-Point



- Same dynamic range as FP32
- Much less precision than FP32

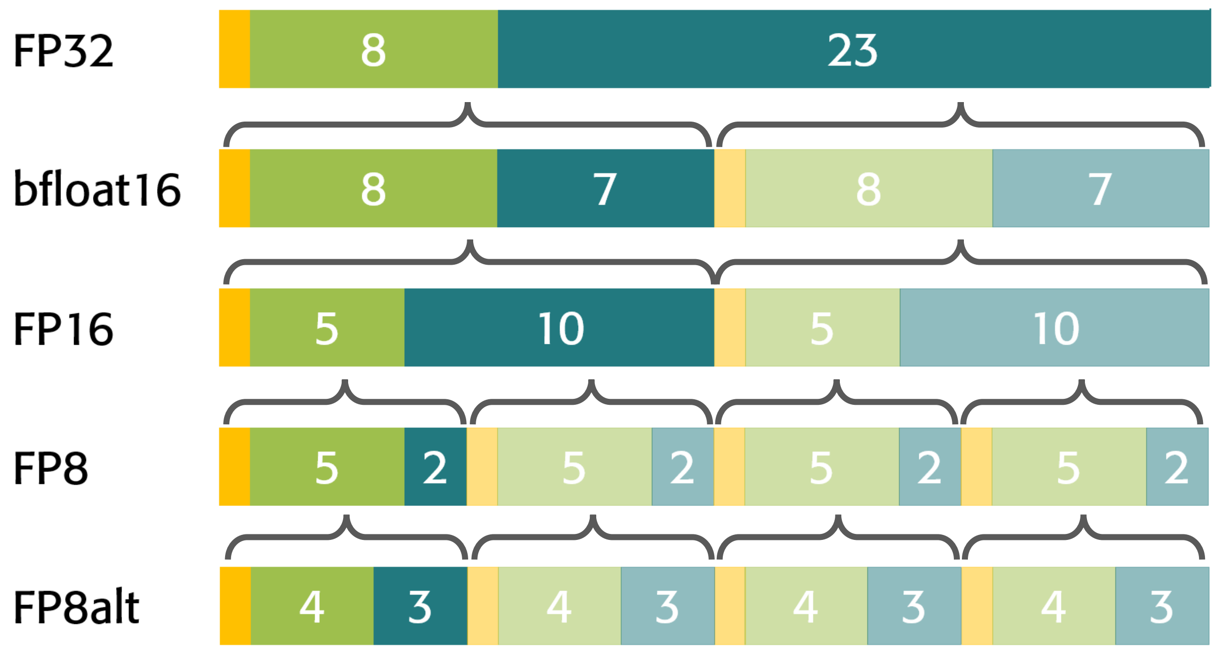
- Less dynamic range than FP32
- Less precision than FP32

- Same dynamic range as FP16
- Less precision than FP16

- Less dynamic range as FP16
- Less precision than FP16

Format	# Representable values	Maximum Value
FP32	4.29×10^{9}	$\approx 3.40 \times 10^{38}$
Bfloat16	65536	$\approx 3.40 \times 10^{38}$
FP16	65536	≈ 65504
FP8	256	≈ 57344
FP8ALT	256	≈ 488

Low-Precision Floating-Point Formats



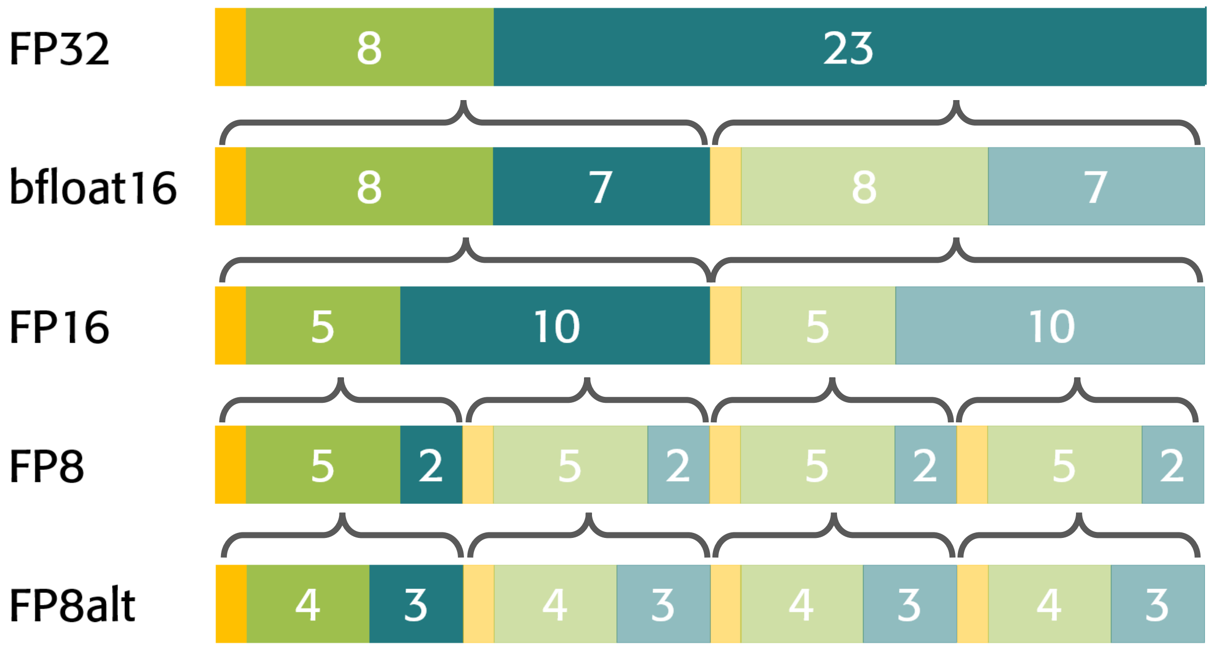
Format	# Representable values	Maximum Value
FP32	4.29×10^9	$\approx 3.40 \times 10^{38}$
Bfloat16	65536	$\approx 3.40 \times 10^{38}$
FP16	65536	≈ 65504
FP8	256	≈ 57344
FP8ALT	256	≈ 488

- However, narrower formats produce lower-accuracy results

- Higher Performance
- Higher energy efficiency
- Lower memory footprint
- Lower data movement energy

- Lower-precision inputs + higher precision accumulation
- + further techniques (SR, scaling)

Low-Precision Floating-Point Formats

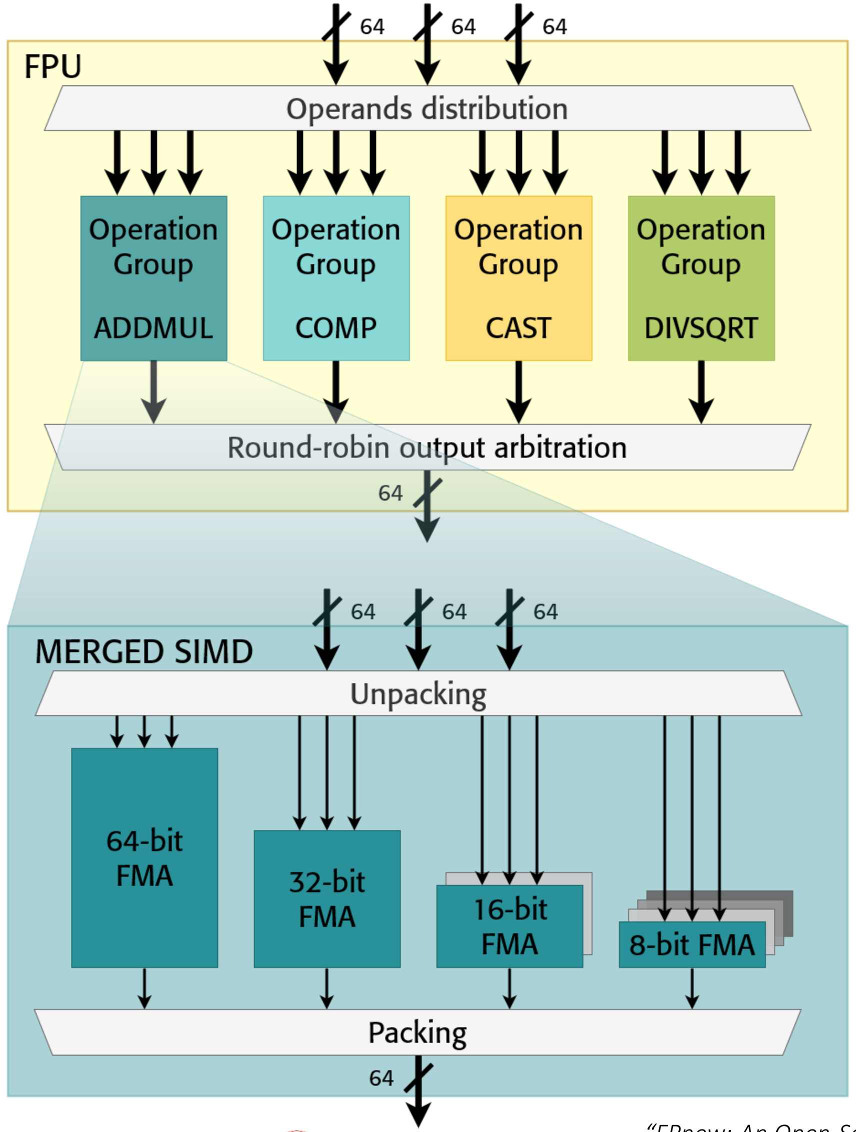


Format	# Representable values	Maximum Value
FP32	4.29×10^9	$\approx 3.40 \times 10^{38}$
Bfloat16	65536	$\approx 3.40 \times 10^{38}$
FP16	65536	≈ 65504
FP8	256	≈ 57344
FP8ALT	256	≈ 488

- However, narrower formats produce lower-accuracy results

Mixed-Precision enables low-precision benefits retaining accuracy

CVFPU: A Multi-Format FPU for Transprecision Computing



- Modular design -- grouped by operation
- Each operation group block is sliced by formats, contains SIMD lanes if SIMD support enabled
- Support for a wide set of formats. Each format can be enabled/disabled
- Each operation group block can be instantiated as:
  - PARALLEL: Each module supports only one format
  - MERGED: Larger-precision modules are reused for lower-precision computations,
  - DISABLED

"FPnew: An Open-Source Multiformat Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing" – S. Mach et al.

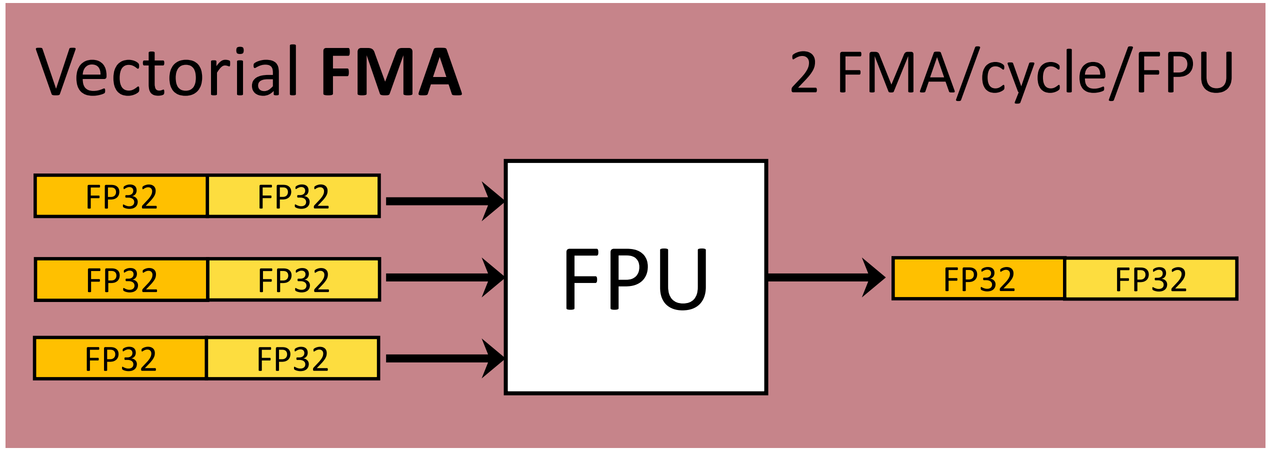
Why Mixed-Precision Instructions?

64-bit FPU with multi-format support

A 32-bit kernel based on FMAs performs 2 FMA/cycle per FPU

A mixed-precision kernel allowing for 16-bit inputs and requiring 32-bit accumulation enables:

- Power savings
- Lower memory footprint



Mixed-Precision FMA vs. Mixed-Precision SDOTP

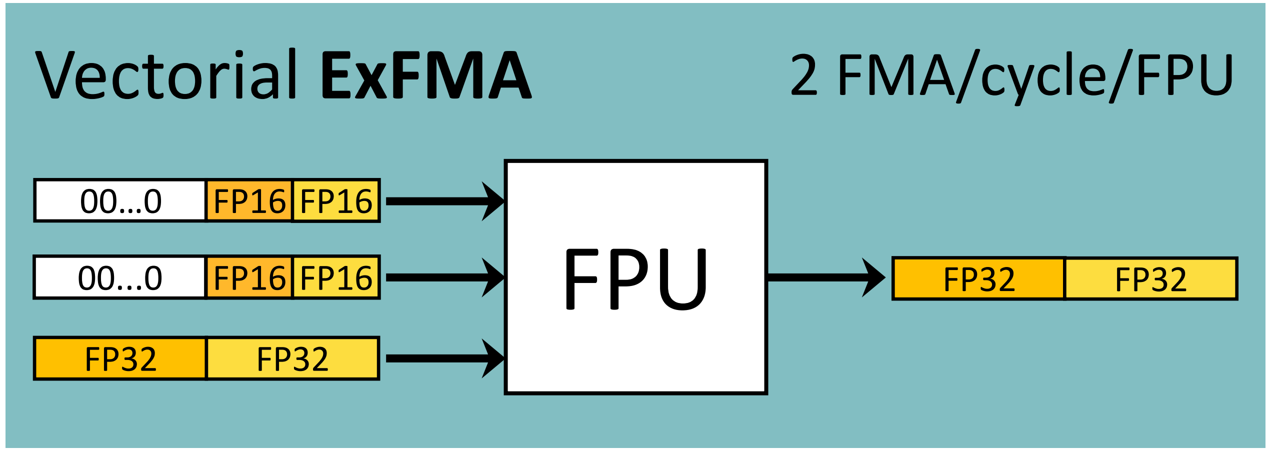
vfwmacc.vv v0, v1, v2

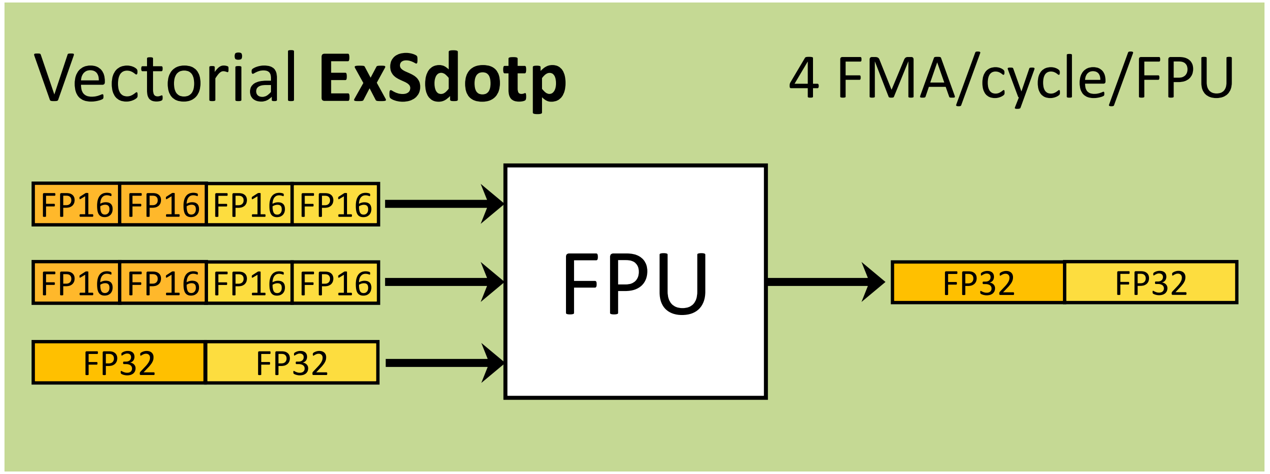
$v0_{2w}[i] \mathrel{+}= v1_w[i] * v2_w[i]$

- Still Expanding FMA **underutilize the FPU bandwidth**
- We can provide the FPU with more data and compute more every cycle → ExSdotp

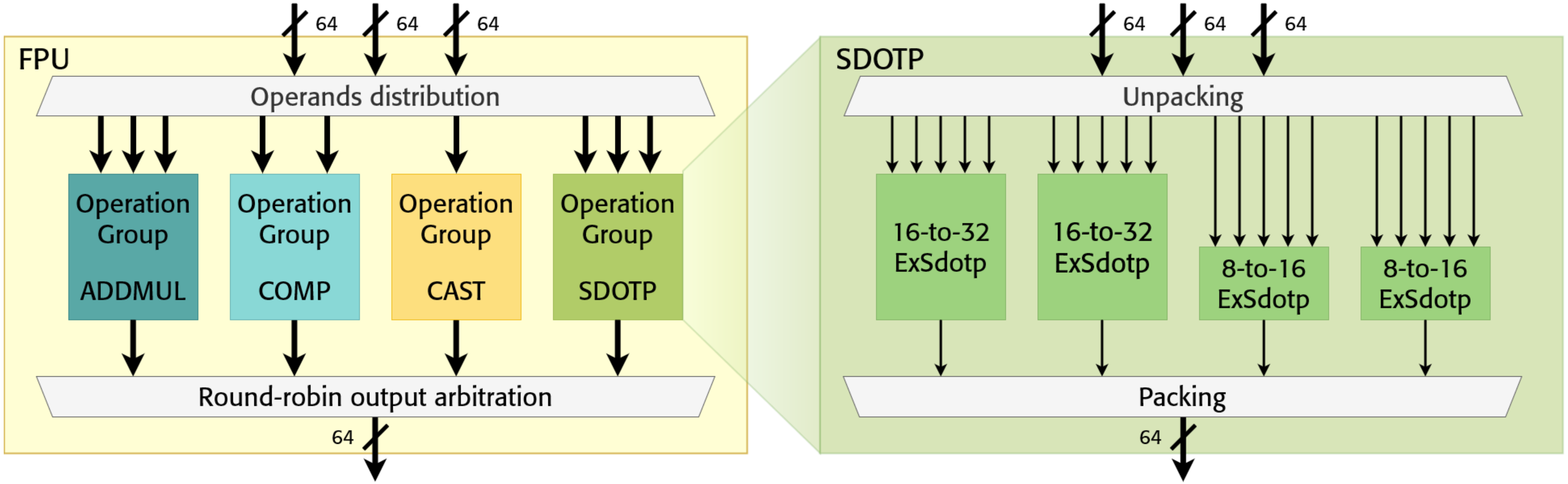
vfwdotp.vv v0,v1,v2

$$v0_{2w}[i] \mathrel{+}= v1_w[2i] * v2_w[2i] + v1_w[2i+1] * v2_w[2i+1]$$





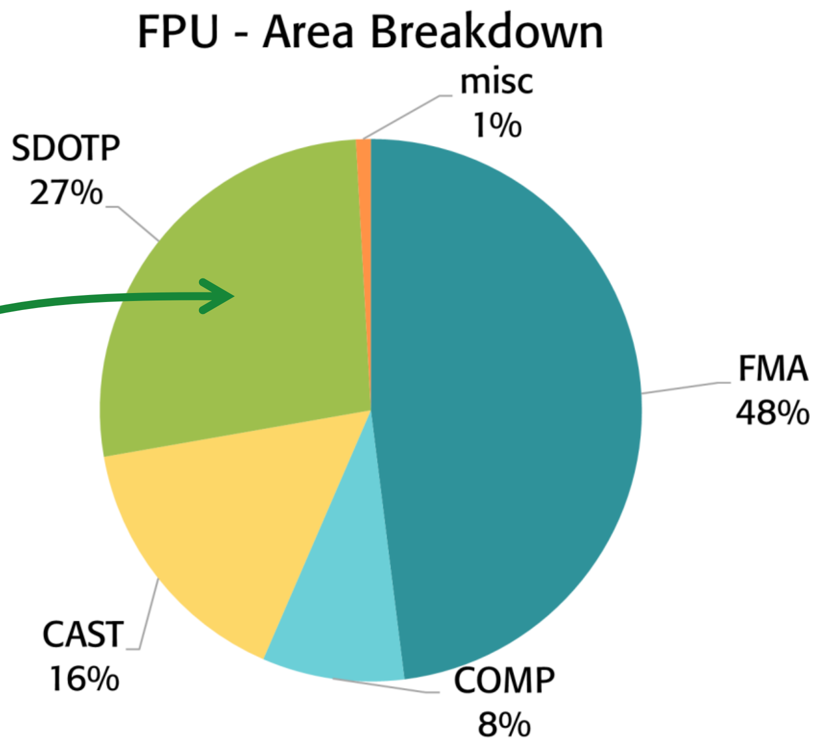
Enhancing CVFPU with SIMD ExSdotp



- **CVFPU** is a highly-parameterized open-source **modular** energy-efficient multi-format FPU
- **SIMD ExSdotp** unit integrated into CVFPU as a new **operation group** block
- SIMD SDOTP: **two** 16-to-32-bit units and **two** 8-to-16-bit units
- Up to **two** 16-to-32-bit ExSdotp and **four** 8-to-16-bit **ExSdotp per cycle**

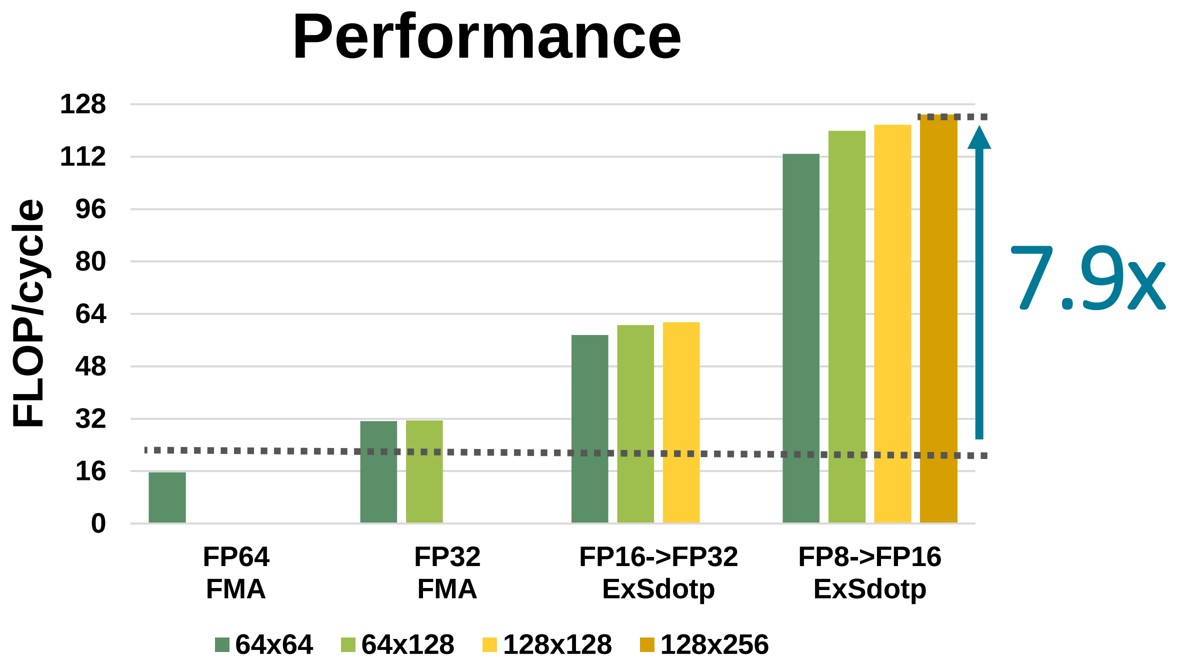
ExSdotp & CVFPU: Area and Timing

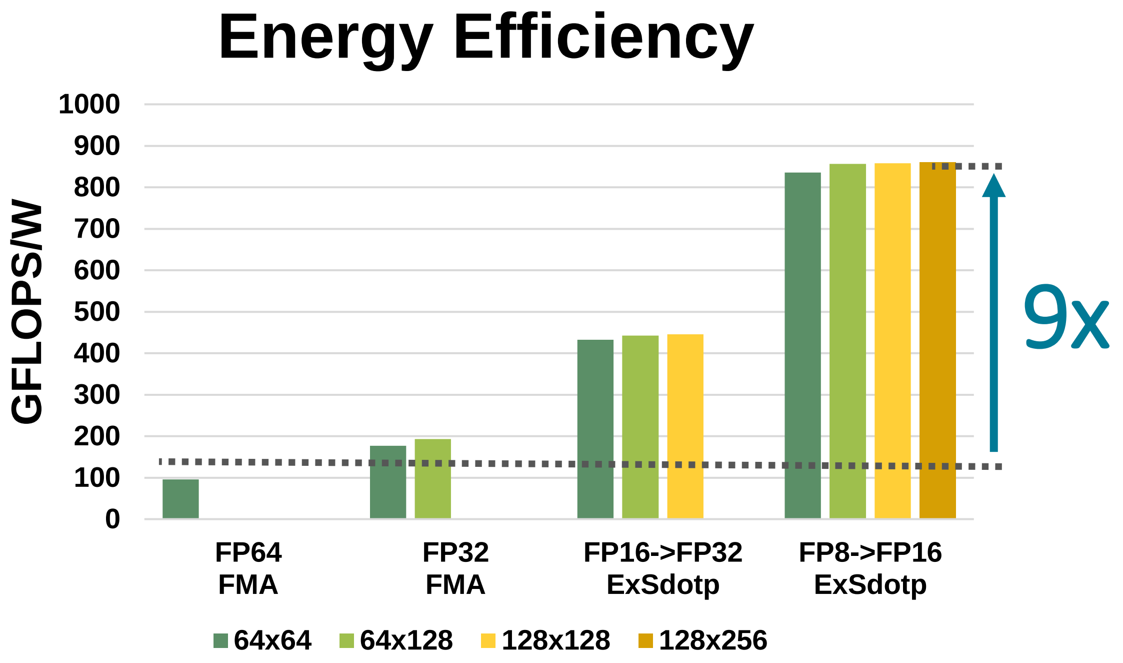
- Implemented in GlobalFoundries 12nm FinFET technology
- Max Frequency → **1.2GHz** (typ 0.8V, 25 °C)
- The SIMD SDOTP unit occupies around 45 kGE, amounting to 27% of the enhanced FPU area (overall FPU area = 165kGE).
- Full extension introduced less than 10% area overhead at a cluster level



ExSdotp Enables Performance & Efficiency Improvements

- **Spatz enhanced with multi-format, mixed-precision FPU**
- 12nm tech, 1 GHz (TT, 0.8V, 25°C)





- Mixed-Precision ExSdotp allows for the same performance as FMA but with higher accumulation precision + around 2x the performance achievable with ExFMAs

Highly-Parametric and Modular: An All-Season FPU

SCALAR CORES

CV32E40P
(RV32F)

CVA6
(RV64FD)

SNITCH
(RV32FD)

VECTOR PROCESSORS

SPATZ
(RVV subset)

ARA
(RVV)

HW ACCELERATORS

RedMulE
(HWPE)

Current Developments

- [Done] New FP32-only DivSqrt unit by T-Head integrated into CVFPU (lower footprint, higher latency with respect to PULP legacy divider)
- [ToDo] Integration of an open-source FP64 DivSqrt unit by T-Head into CVFPU
- [Ongoing] Precision tracking and recovery enabled by mixed-precision ISAs (ad-hoc higher-precision computations upon the detection of precision losses)

Open-Source Implementation and Publications

github.com/pulp-platform/cvfpu

“MiniFloat-NN and ExSdotp: An ISA Extension and a Modular Open Hardware Unit for Low-Precision Training on RISC-V Cores”, L. Bertaccini et al. (ARITH22)

Submitted journal paper extension including MiniFloat results on vector processors soon available on arXiv