PULP PLATFORM

Open Source Hardware, the way it should be!

Working with RISC-V

Part 4 of 5 : PULP Extensions and Accelerators

Davide Rossi <davide.rossi@unibo.it>

Luca Benini <luca.benini@unibo.it>

ETH zürich

Summary

- Part 1 – Introduction to RISC-V ISA
- Part 2 – Advanced RISC-V Architectures
- Part 3 – PULP concepts
- **Part 4 – PULP Extensions and Accelerators**
  - ISA Extensions
  - Shared Memory Accelerators
- Part 5 – PULP based chips

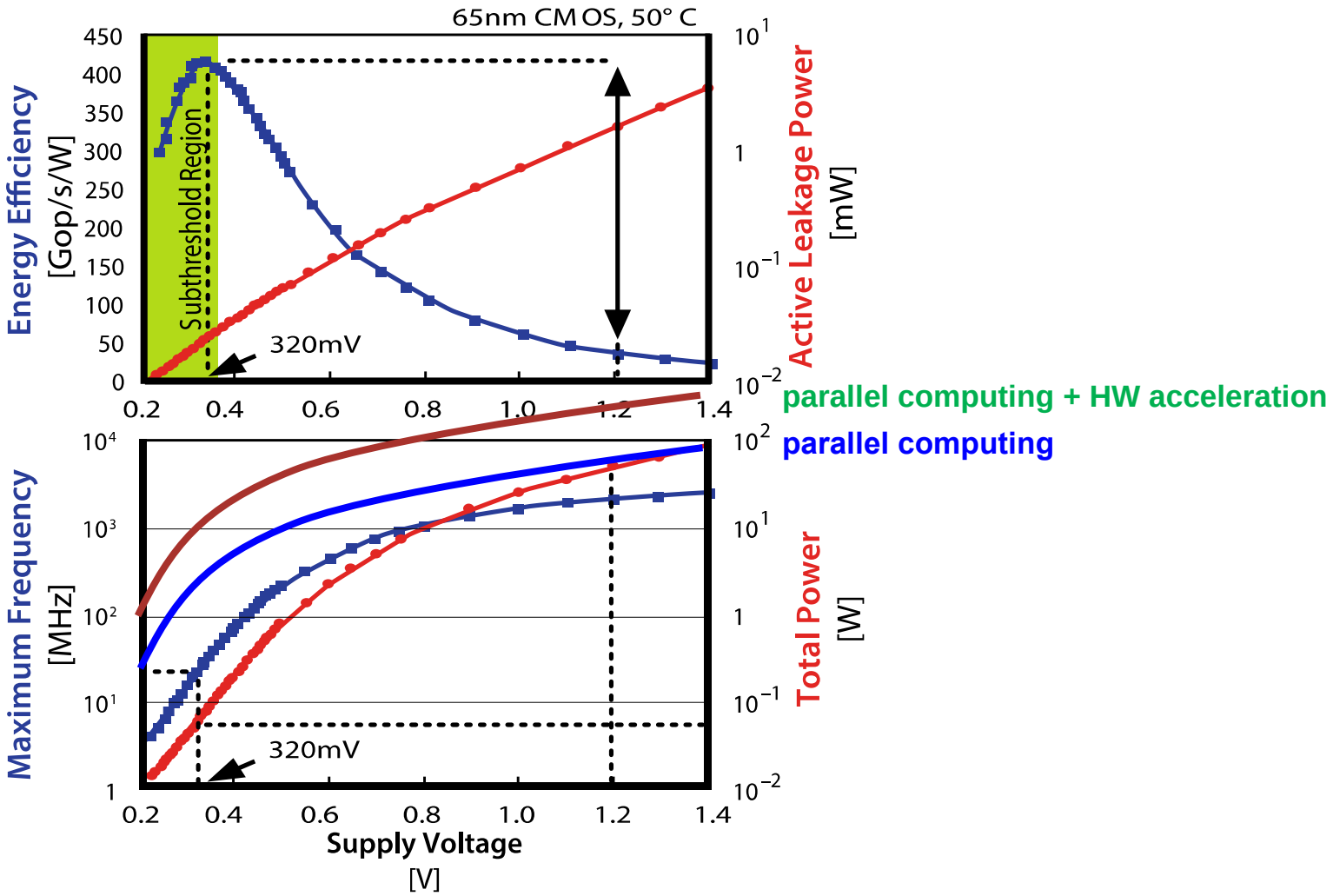
Why Hardware Acceleration, really?

- **PULP ♡ Software** (yes, even the hardware designers!)…
  - PULP has been designed as a ***programmable***, ***software-oriented*** platform
  - Software is ***flexible*** and SW-programmable platforms are capable of dealing with applications that are ***highly irregular*** or ***unexpected*** at design time
  - Software can be ***highly efficient***, when it's fully using features exposed by the hardware
  - Software code is accessible by ***many more developers***: intrinsically more open

- … but software is **not always enough**, and we also need **accelerators**
  - Some applications have ***too stringent constraints*** of energy/power
  - Some kernels are ***often used*** and ***computationally heavy*** at the same time
  - **Too many cores** (= area **= $$$**) would be required to meet the performance constraints
  - Different kinds of accalerators are needed to address different needs in terms of **Performance**, **Flexibility**, **Area**.

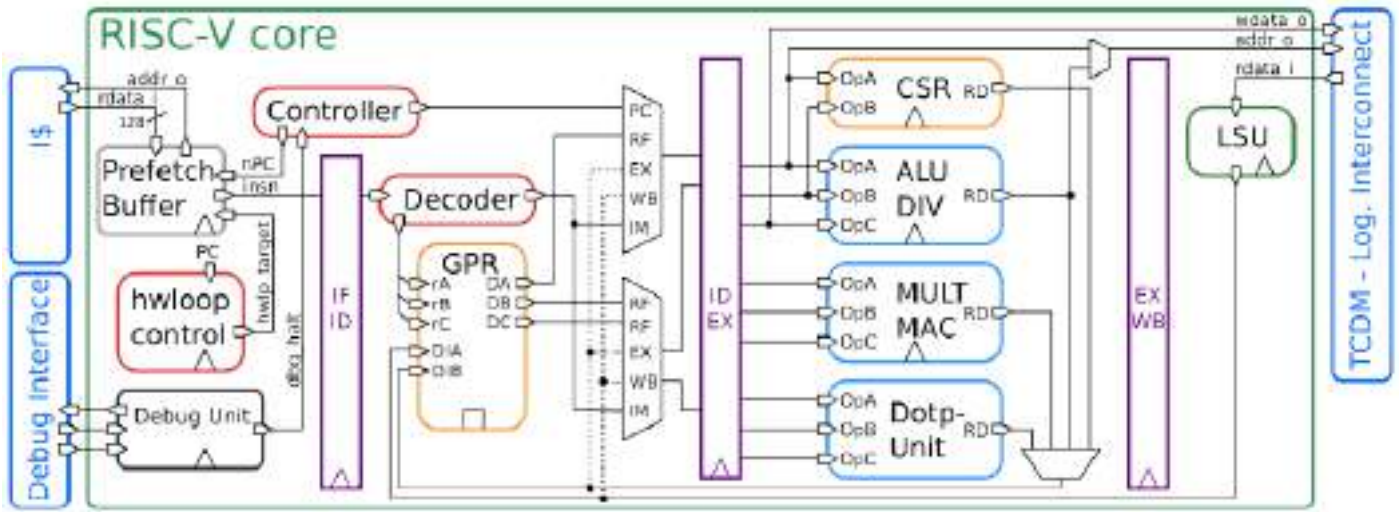
Hardware Accelerators from a PULP-y Perspective

Adapted from Borkar and Chien, The Future of Microprocessors, *Communications of the ACM, May 2011*



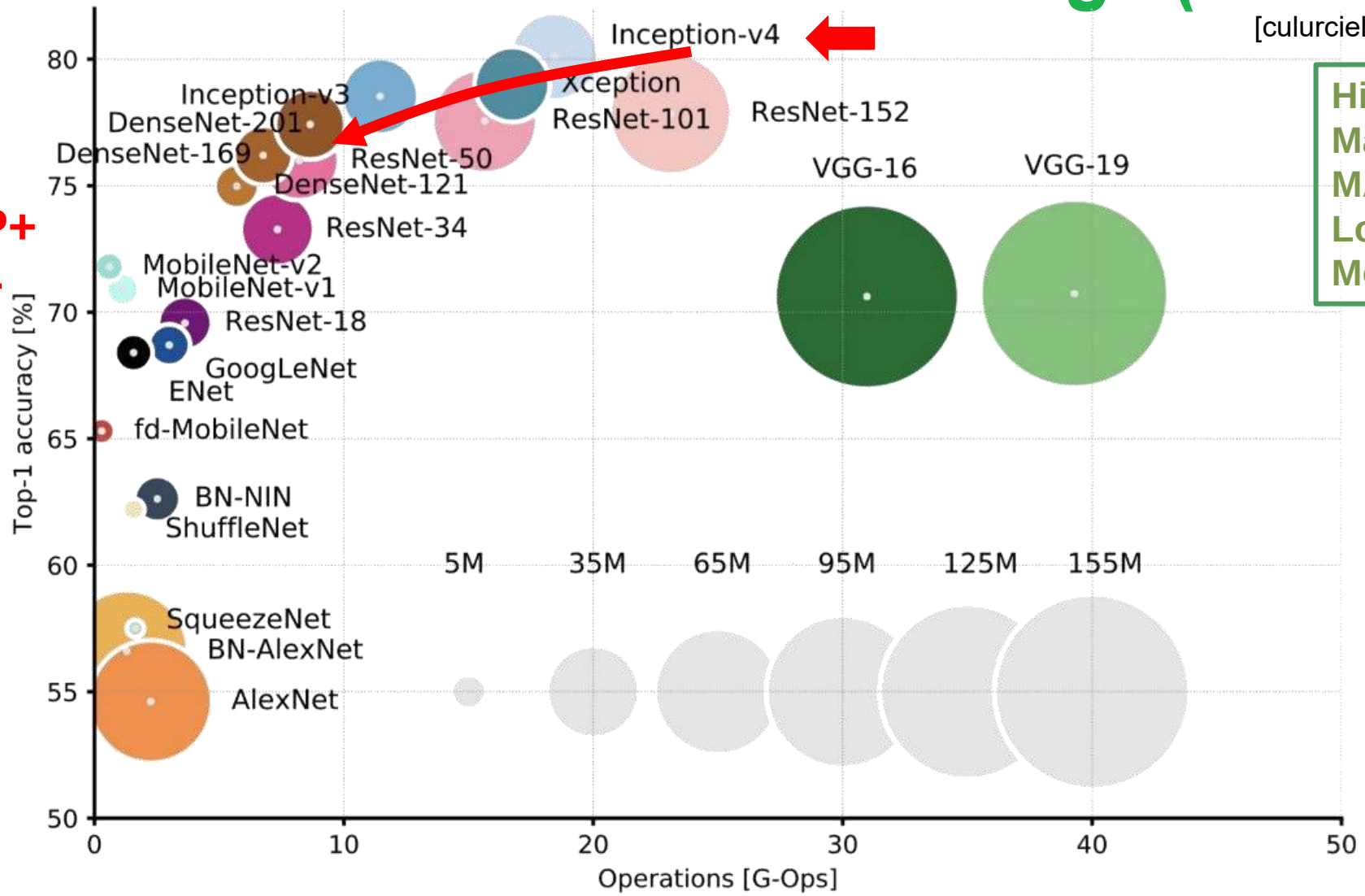
Application Specific Instruction Processors

- Integrated in the Pipeline of Processors (ID Stage, EX Stage, WB Stage)
- Suffer from Register File Bandwidth Bottleneck (only 2 operands…)
- Require Adapting Compiler and Binutils
- Low Overhead for Control
- Auxiliary Processing Units (APU) Interface available in the RI5CY



AI Workloads from Cloud to Edge (Extreme?)

[culurciello18]

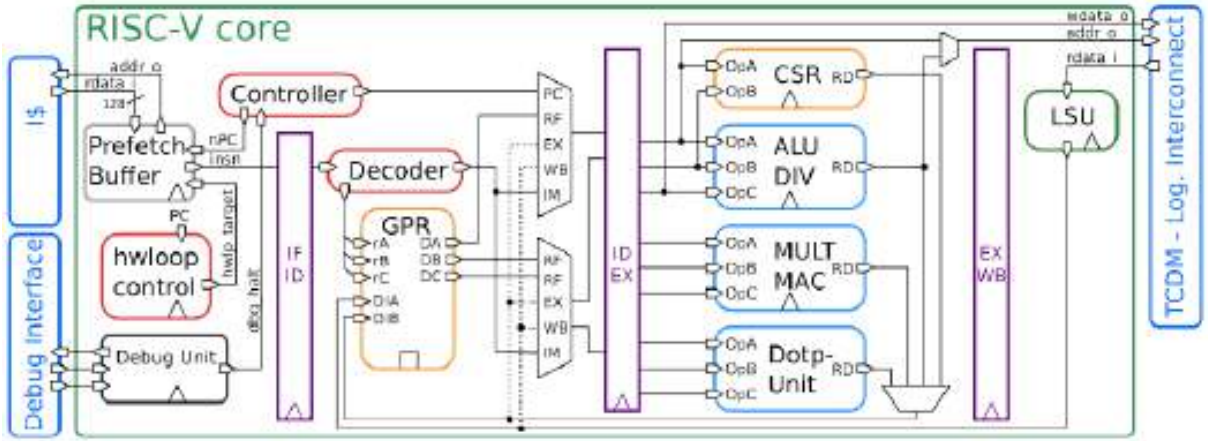


GOP+
MB+

- High OP/B ratio
- Massive Parallelism
- MAC-dominated
- Low precision OK
- Model redundancy

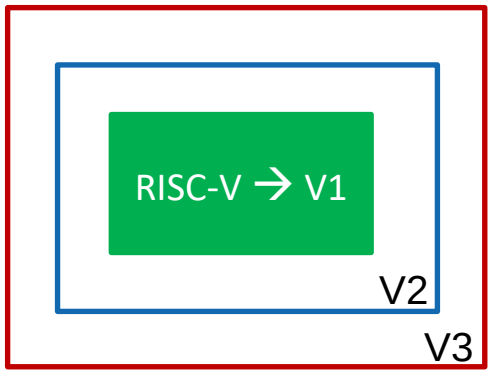
RI5CY – Recap from Part 1

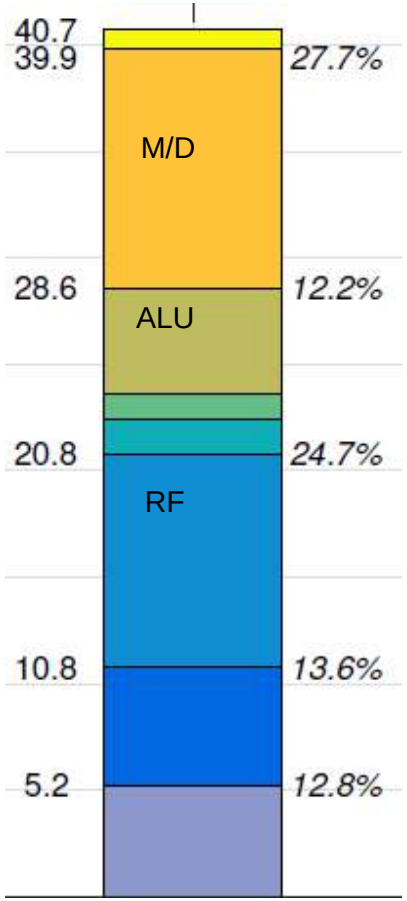
3-cycle ALU-OP, 4-cyle MEM-OP→IPC loss: LD-use, Branch



40 kGE
70% RF+DP

RISC-V core





V1 Baseline RISC-V RV32IMC (not good for ML)

V2 HW loops, Post modified Load/Store, Mac

V3 SIMD 2/4 + DotProduct + Shuffling
Bit manipulation, Lightweight fixed point

XPULP 25 kGE → 40 kGE (1.6x) but 9+ times DSP!

PULP-NN: Xpulp ISA exploitation

8-bit Convolution

RV32IMC

N

```
addi  a0,a0,1
addi  t1,t1,1
addi  t3,t3,1
addi  t4,t4,1
lbu   a7,-1(a0)
lbu   a6,-1(t4)
lbu   a5,-1(t3)
lbu   t5,-1(t1)
mul   s1,a7,a6
mul   a7,a7,a5
add   s0,s0,s1
mul   a6,a6,t5
add   t0,t0,a7
mul   a5,a5,t5
add   t2,t2,a6
add   t6,t6,a5
bne   s5,a0,1c000bc
```

RV32IMCXpulp

N/4

```
lp.setup
p.lw  w1, 4(a0!)
p.lw  w2, 4(a1!)
p.lw   x1, 4(a2!)
p.lw   x2, 4(a3!)
pv.sdotsp.b   s1, w1, x1
pv.sdotsp.b   s2, w1, x2
pv.sdotsp.b   s3, w2, x1
pv.sdotsp.b   s4, w2, x2
end
```

HW Loop

LD/ST with post increment

8-bit SIMD sdotp

9x less instructions than RV32IMC

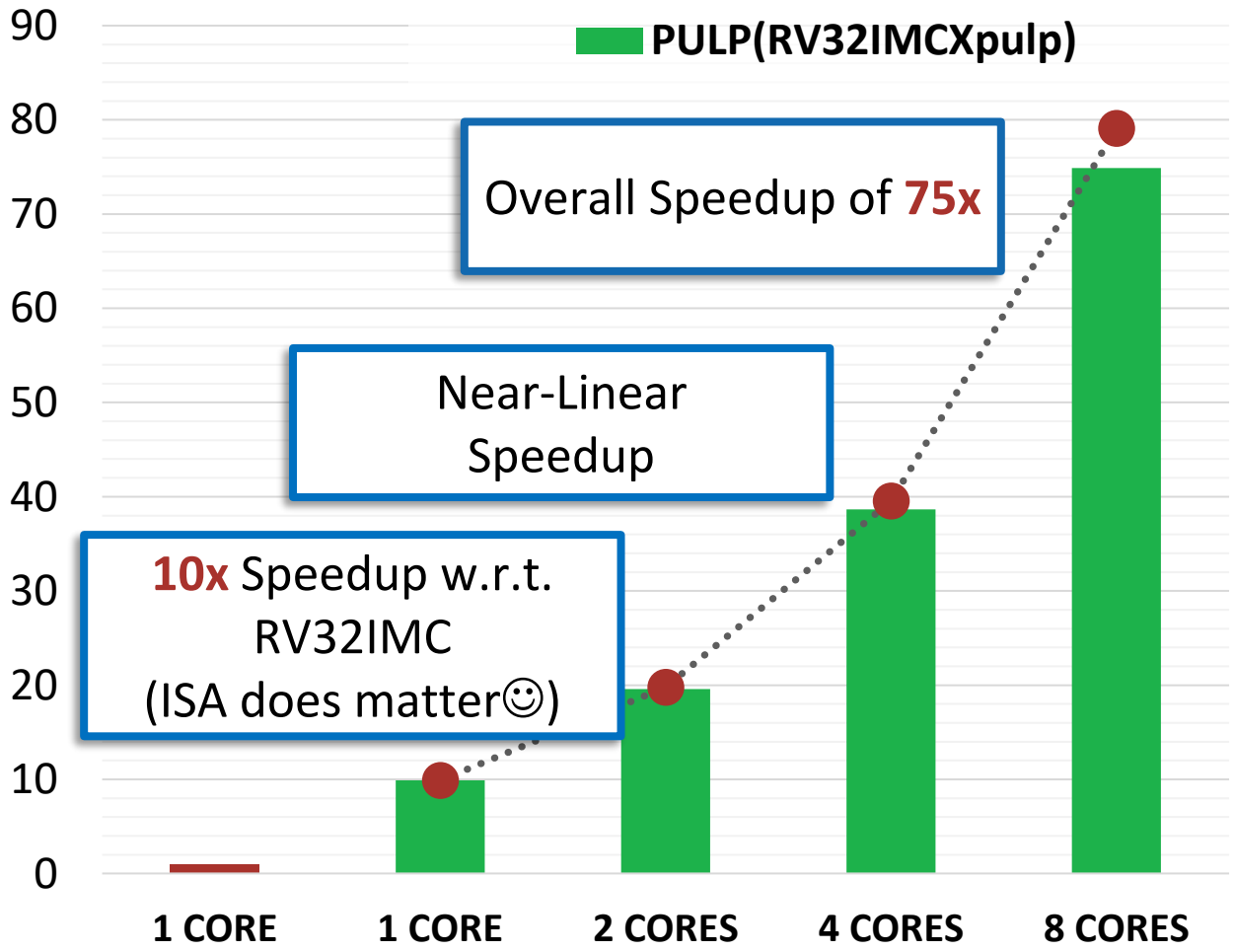
Pooling & ReLu

HW loop
LD/ST with post-increment
8-bit SIMD max, avg

Garofalo, Angelo et al. "PULP-NN: Accelerating Quantized Neural Networks on Parallel Ultra-Low-Power RISC-V Processors." Philosophical Transactions of the Royal Society A

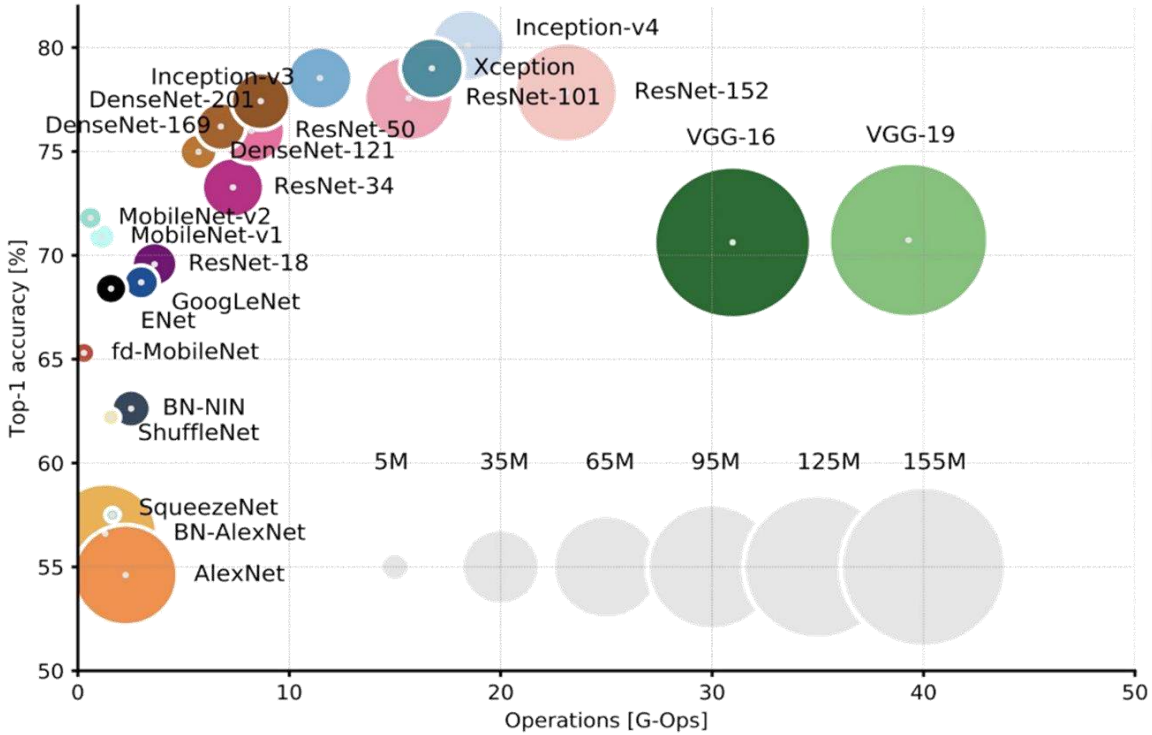
Results: RV32IMCXpulp vs RV32IMC

- 8-bit convolution
  - Open source DNN library
- 10x through xPULP
  - Extensions bring real speedup
- Near-linear speedup
  - Scales well for regular workloads.
- ~2 8-bit MAC/Cycle/core
- 75x overall gain



Pushing Further: Quantized Neural Networks

- Model



- Quantization (*)

Quantization Method	Top1 Accuracy		Weight Memory Footprint	
Full-Precision	70.9%		16.27 MB	
INT-8	70.1%	**0.8%**	4.06 MB	**4x**
INT-4	66.46%	**4.4%**	2.35 MB	**7x**
Mixed-Precision	68%	**2.9%**	2.09 MB	**8x**

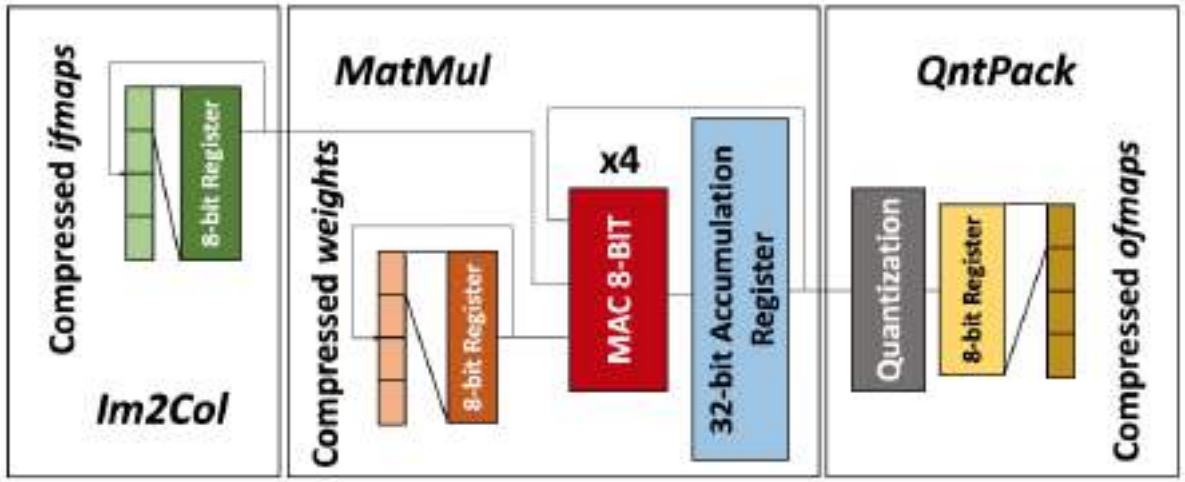
Courtesy of Rusci M. «Example on MobilenetV1_224_1.0.»C

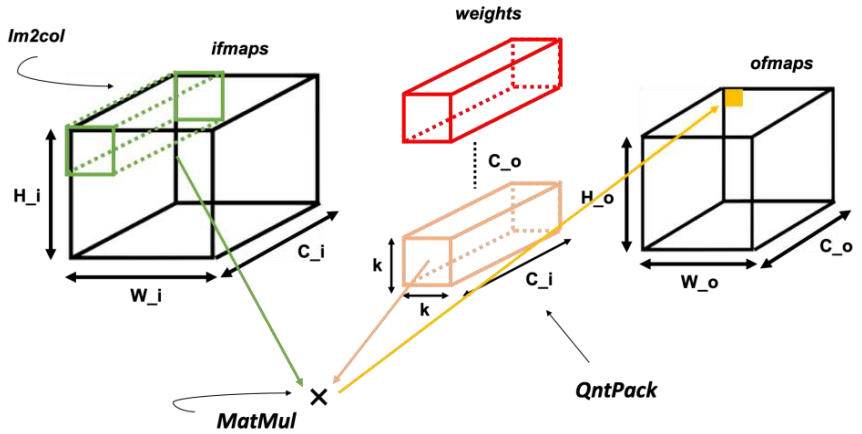
Quantized Neural Networks (QNNs) are the natural target for execution on constrained edge platforms.

Rusci M. et al., Memory-Driven Mixed Low Precision Quantization For Enabling Deep Network Inference On Microcontrollers. arXiv preprint arXiv:1905.13082.

PULP-NN Convolution Kernels (8-bit)

Example of quantized kernel structure



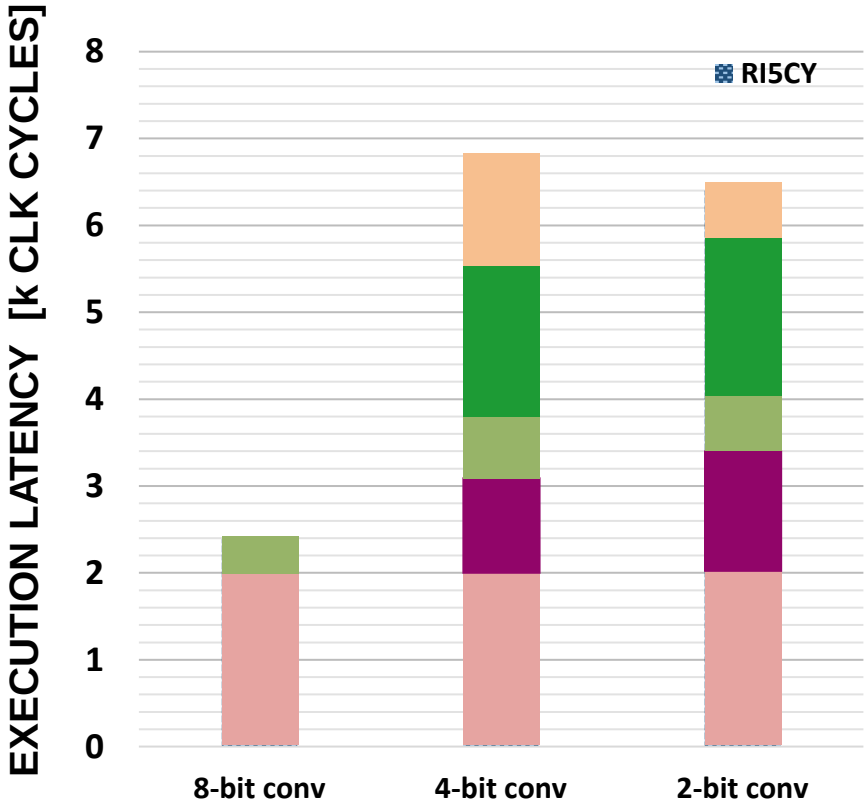


- Leverages an optimized computational model based on CMSIS-NN library
- Exploits HWC organization + SIMD MAC of Xpulpv2 ISA extension

- At every MatMul iteration it fetches
  - 2 im2col
  - 4 filters
- And generates 8 output pixels

XpulpV2 Overheads (2-bit, 4-bit kernels)

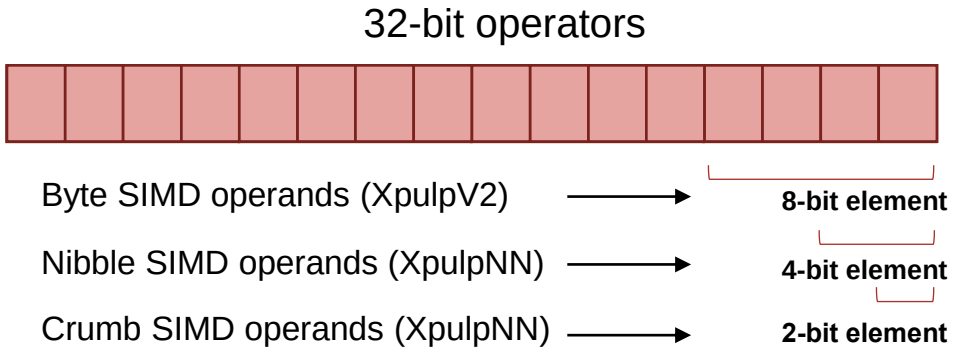
Sub-byte Convolution Kernel on XpulpV2 ISA



- 8-bit MatMul kernel
- Overhead to extract compressed weights
- Im2col function
- Overhead to extract compressed ifmap
- Re-Quantization

XpulpNN: ISA Extensions for QNN on PULP

Arithmetic SIMD instructions



Supported Ops: ALU, Comparison, Shift, abs, Dot Product

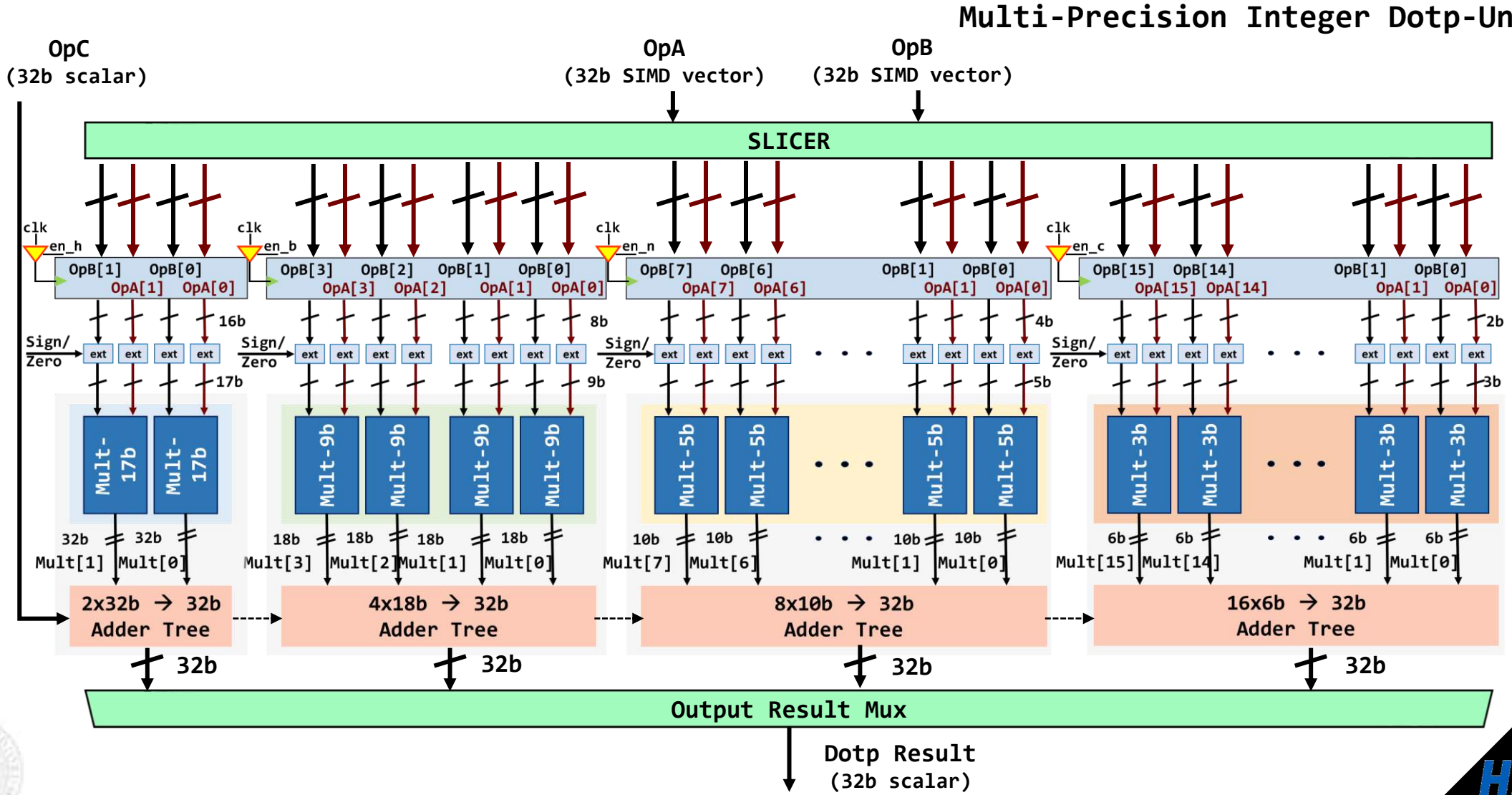
No need to unpack sub-byte data

Multi-cycle instruction to efficiently handle the quantization process in HW

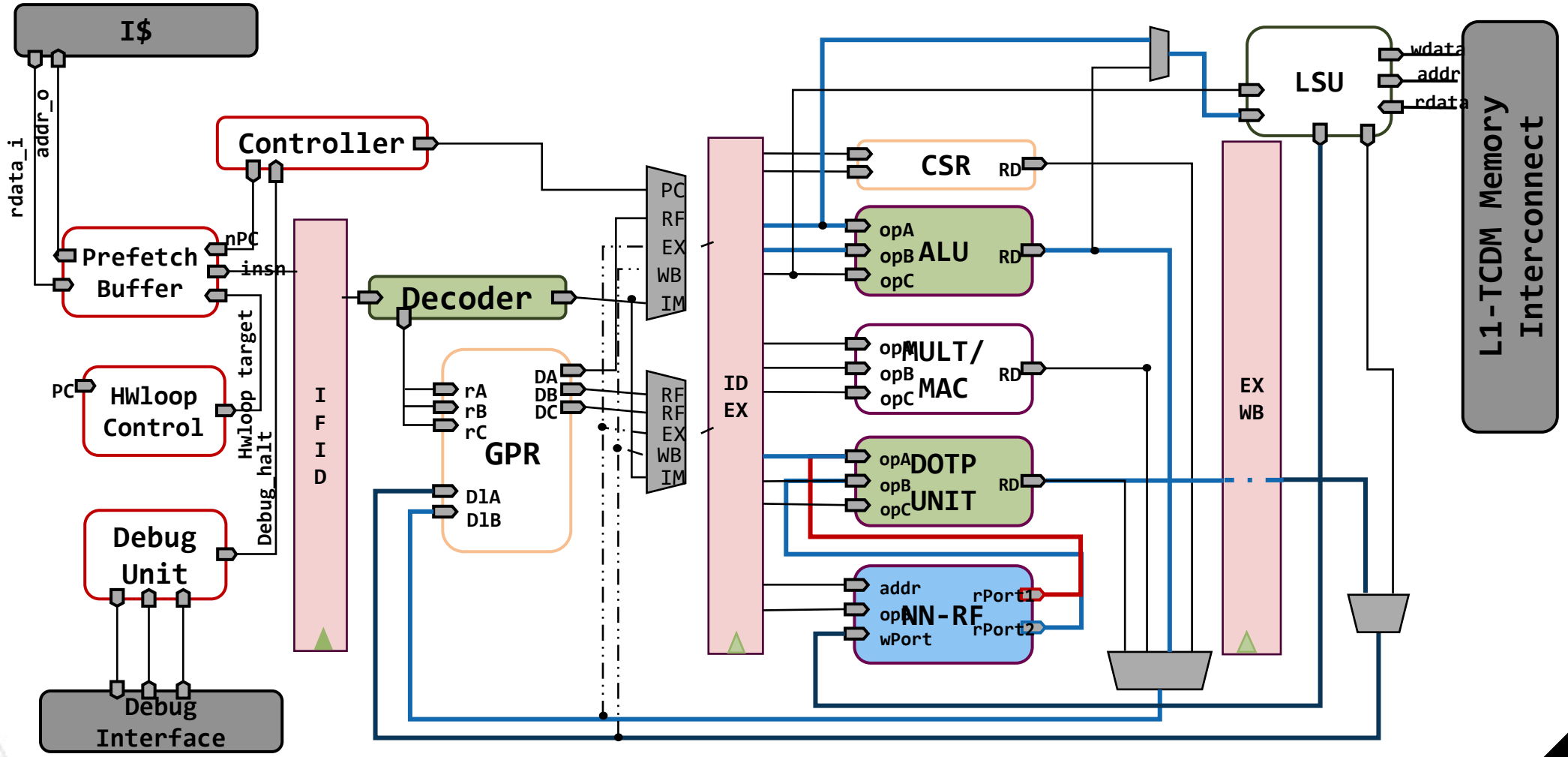
ALU SIMD Op.	Description for *nibble*
pv.add[.sc].{n, c}	rD[i] = rs1[i] + rs2[i]
pv.sub[.sc].{n, c}	rD[i] = rs1[i] - rs2[i]
pv.avg(u)[.sc].{n, c}	rD[i] = (rs1[i] + rs2[i])>>1
Vector Comparison Op.	
pv.max(u)[.sc].{n, c}	rD[i] = rs1[i] > rs2[i] ? rs1[i] : rs2[i]
pv.min(u)[.sc].{n, c}	rD[i] = rs1[i] < rs2[i] ? rs1[i] : rs2[i]
Vector Shift Op.	
pv.srl[.sc].{n, c}	rD[i] = rs1[i] >> rs2[i] Shift is logical
pv.sra[.sc].{n, c}	rD[i] = rs1[i] >> rs2[i] Shift is arithmetic
pv.sll[.sc].{n, c}	rD[i] = rs1[i] << rs2[i]
Vector abs Op.	
pv.abs.{n, c}	rD[i] = rs1[i] < 0 ? -rs1[i] : rs1[i]
Dot Product Op.	
pv.dotup[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7]
pv.dotusp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7]
pv.dotsp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7]
pv.sdotup[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7] + rD
pv.sdotusp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7] + rD
pv.sdotsp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7] + rD
Quantization Op.	Dedicated Quantization instruction
pv.qnt.{n, c}	

A. Garofalo, G. Tagliavini, F. Conti, L. Benini and D. Rossi, "XpulpNN: Enabling Energy Efficient and Flexible Inference of Quantized Neural Networks on RISC-V based IoT End Nodes," in *IEEE Transactions on Emerging Topics in Computing, 2021*

1) Multi-precision Dotp Unit



2) MAC/Load Extension



Mac-Load: PULP-NN inner kernel

8-bit *Matmul kernel* with PULP-NN

```
      lp.setup      l1, l2,end
      p.lw          w1, 4(aw1!)
      p.lw          w2, 4(aw2!)
      p.lw          w3, 4(aw3!)
      p.lw          w4, 4(aw4!)
      p.lw          x1, 4(ax1!)
      p.lw          x2, 4(ax2!)
      pv.sdotusp.b  s1, x1, w1
      pv.sdotusp.b  s2, x1, w2
      pv.sdotusp.b  s3, x1, w3
      pv.sdotusp.b  s4, x1, w4
      pv.sdotusp.b  s5, x2, w1
      pv.sdotusp.b  s6, x2, w2
      pv.sdotusp.b  s7, x2, w3
end:  pv.sdotusp.b  s8, x2, w4
```

HW LOOP

LD/ST WITH POST INCREMENT

8-B SIMD MAC

8 SIMD MACs with 6 explicit LOADs

8-bit *MatMul kernel* With MAC + Load

INIT NN-RF

```
      pv.nnsdotusp.h  zero, aw1,16
      pv.nnsdotusp.h  zero, aw2,18
      pv.nnsdotusp.h  zero, aw3,20
      pv.nnsdotusp.h  zero, aw4,22
      pv.nnsdotusp.h  zero, ax1,8
      lp.setup        l1, l2, end
      pv.nnsdotup.h   zero,ax2,9
      pv.nnsdotusp.b  s1, aw2, 0
      pv.nnsdotusp.b  s2, aw4, 2
      pv.nnsdotusp.b  s3, aw3, 4
      pv.nnsdotusp.b  s4, ax1, 14
      pv.nnsdotusp.b  s5, aw2, 17
      pv.nnsdotusp.b  s6, aw4, 19
      pv.nnsdotusp.b  s7, aw3, 21
end:  pv.nnsdotusp.b  s8, aw1, 23
```

8 SIMD MACs with 1 explicit LOAD

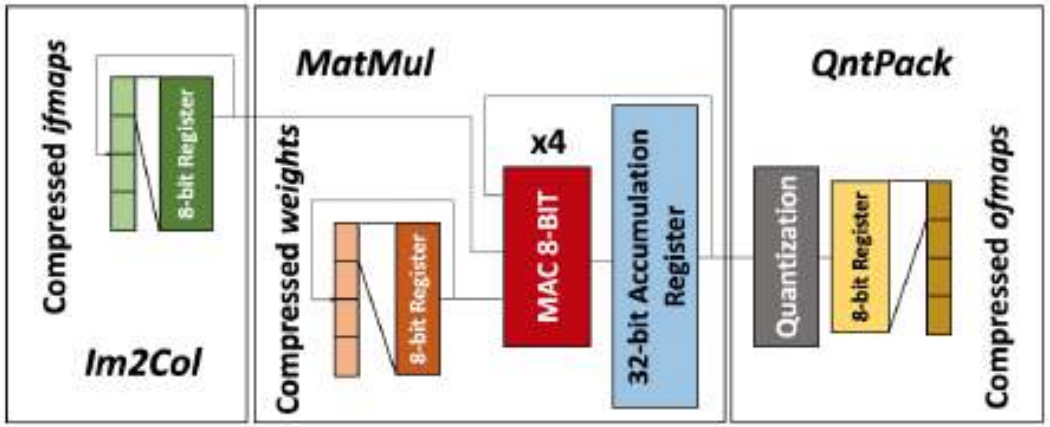
W_i X_i : weight/activation elements

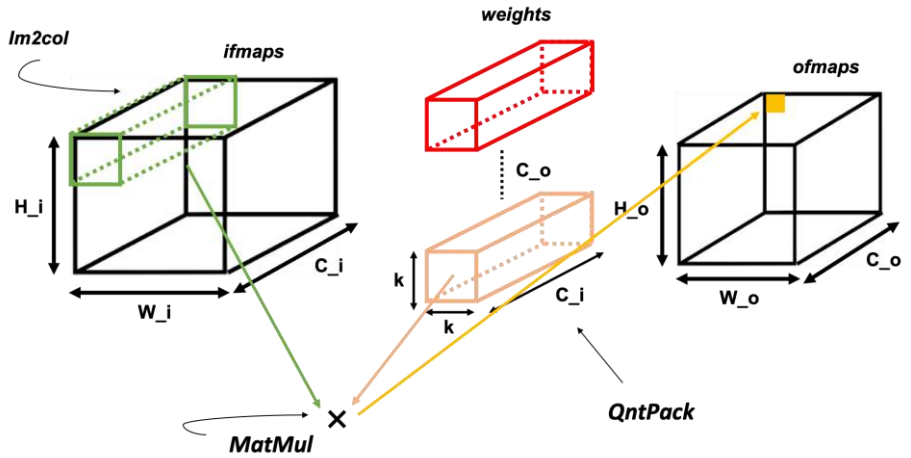
AW_i AX_i : addresses for the MEM access

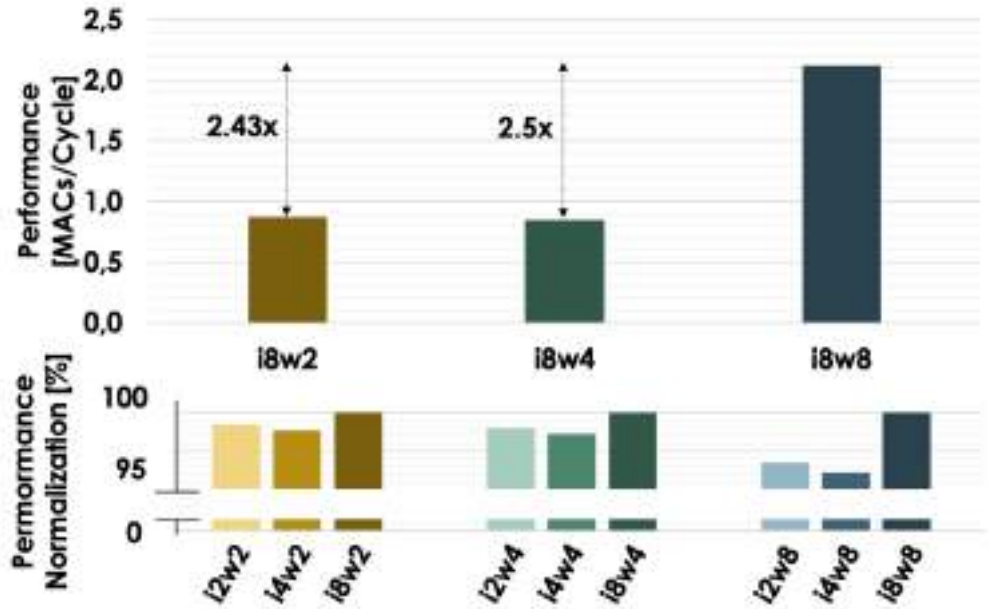
S_i : accumulators

L_i : loop setup

Mixed-Precision Kernel (Overheads)



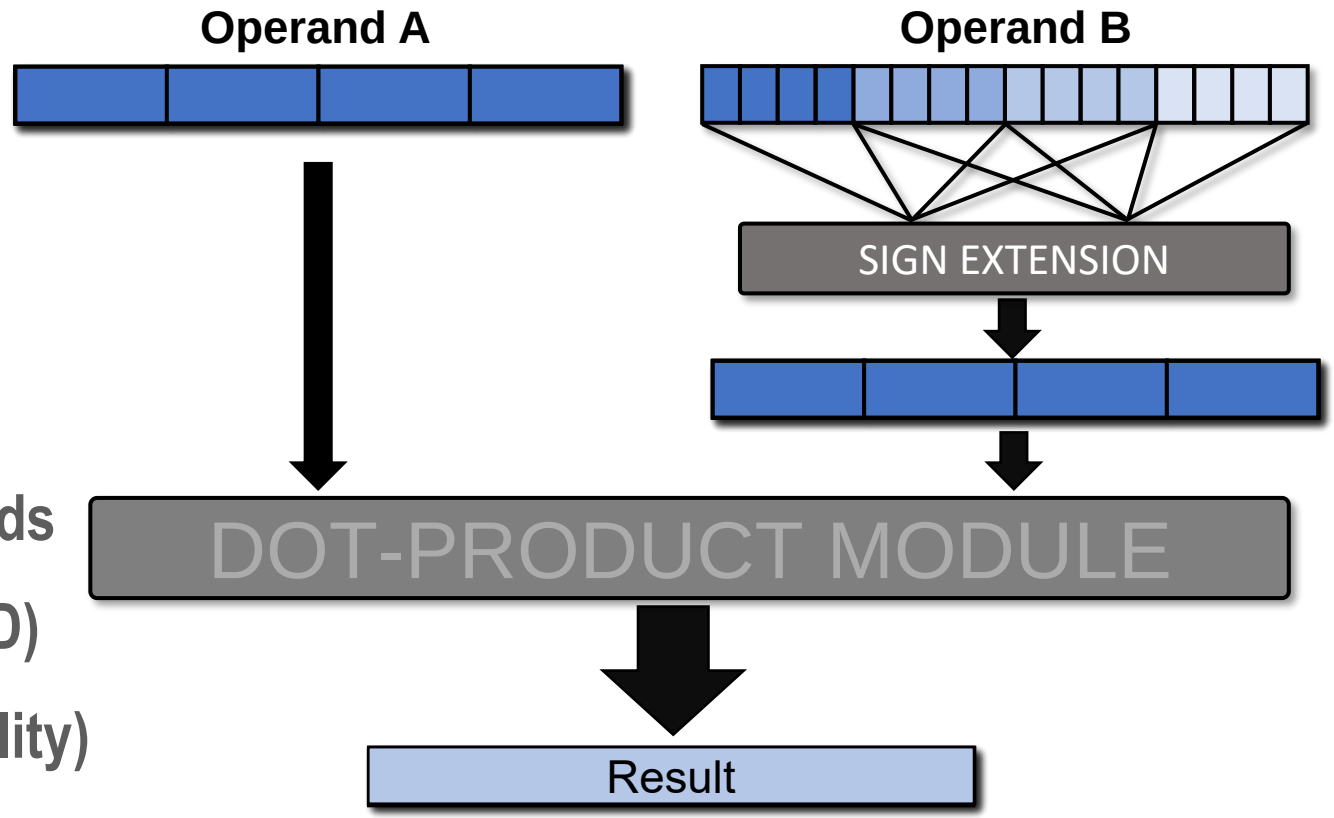




N.Bruschi et. al., "Enabling mixed-precision quantized neural networks in extreme-edge devices," ACM International Conference on Computing Frontiers, 2020.

Mixed Precision SIMD Processor

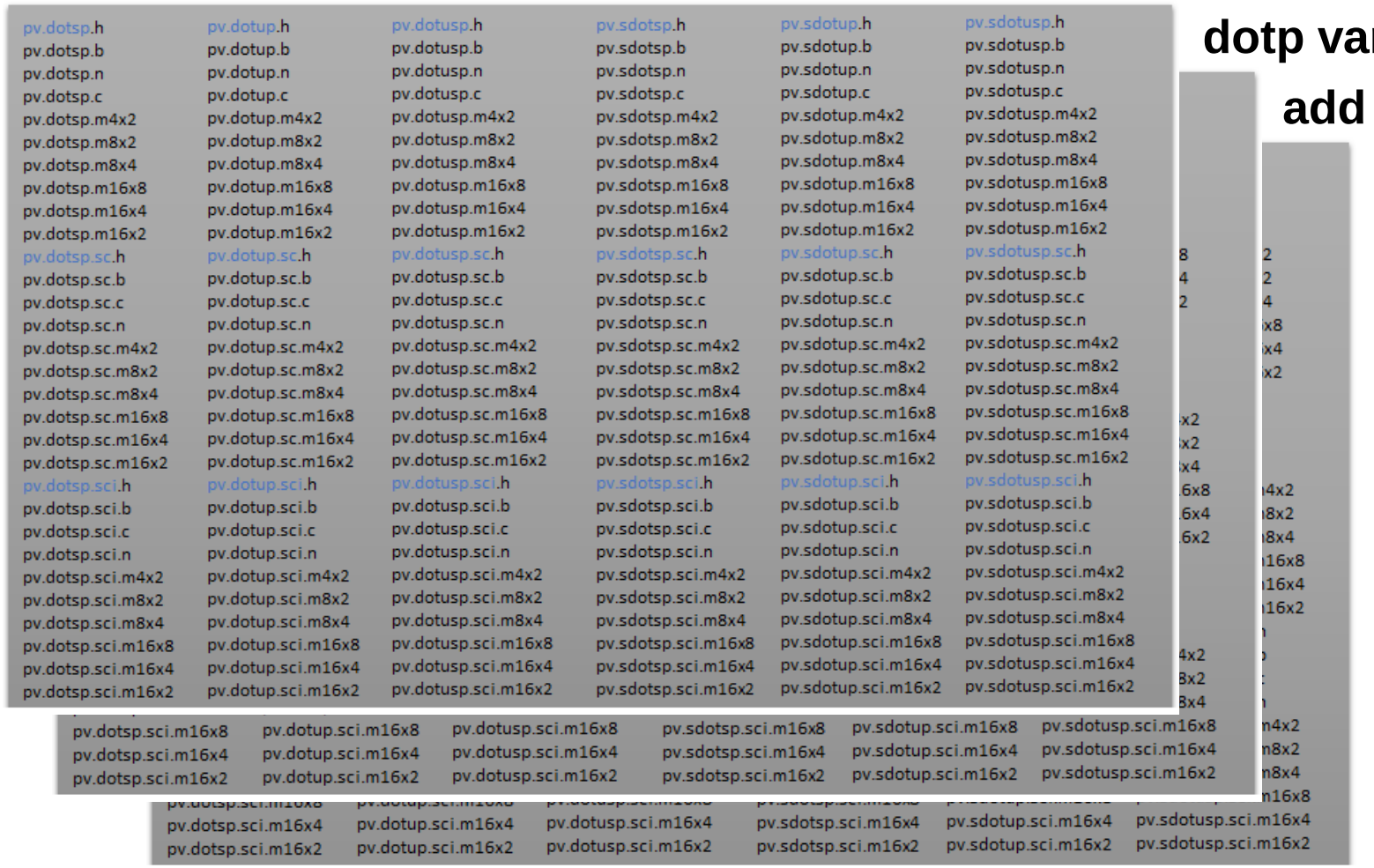
- Can support all variants:
  - 16x16, 16x8, 16x4, 16x2
  - 8x8, 8x4, 8x2
  - 4x4, 4x2
  - 2x2
- Avoids Pack/unpack Overheads
- Maximized performance (SIMD)
- Maximizes RF use (Data Locality)



How to encode all these instructions?

G. Ottavi, A. Garofalo, G. Tagliavini, F. Conti, L. Benini and D. Rossi, "A Mixed-Precision RISC-V Processor for Extreme-Edge DNN Inference," 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2020, pp. 512-517

Mixed-Precision Core: New Formats Required



dotp variants

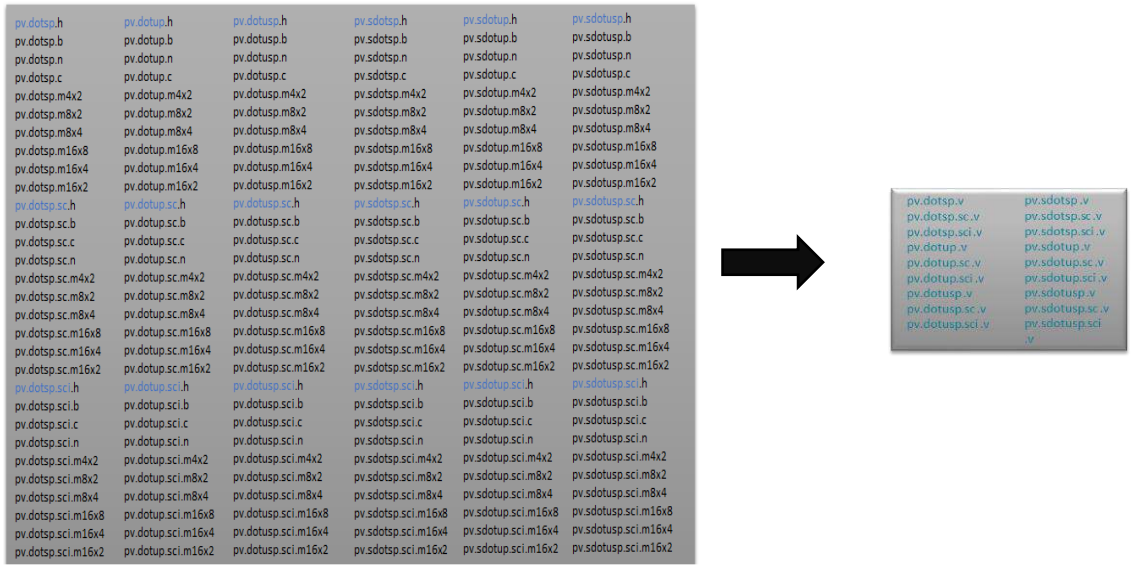
add variants

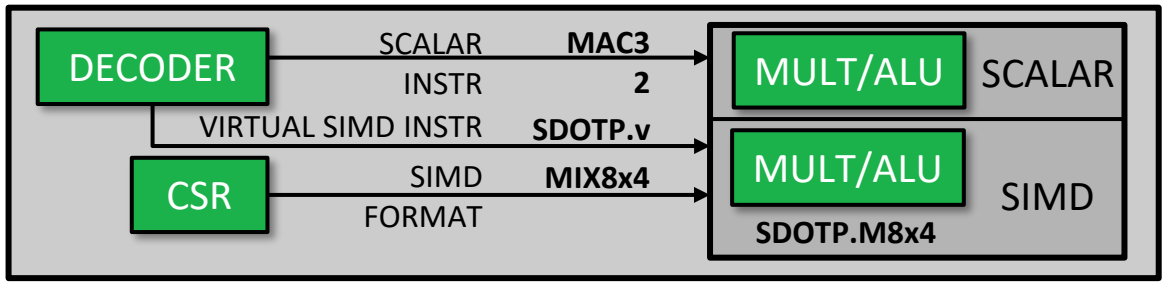
sub variants
avg variants
shift variants
max variants
min variants
abs variants
…

> 500 instructions

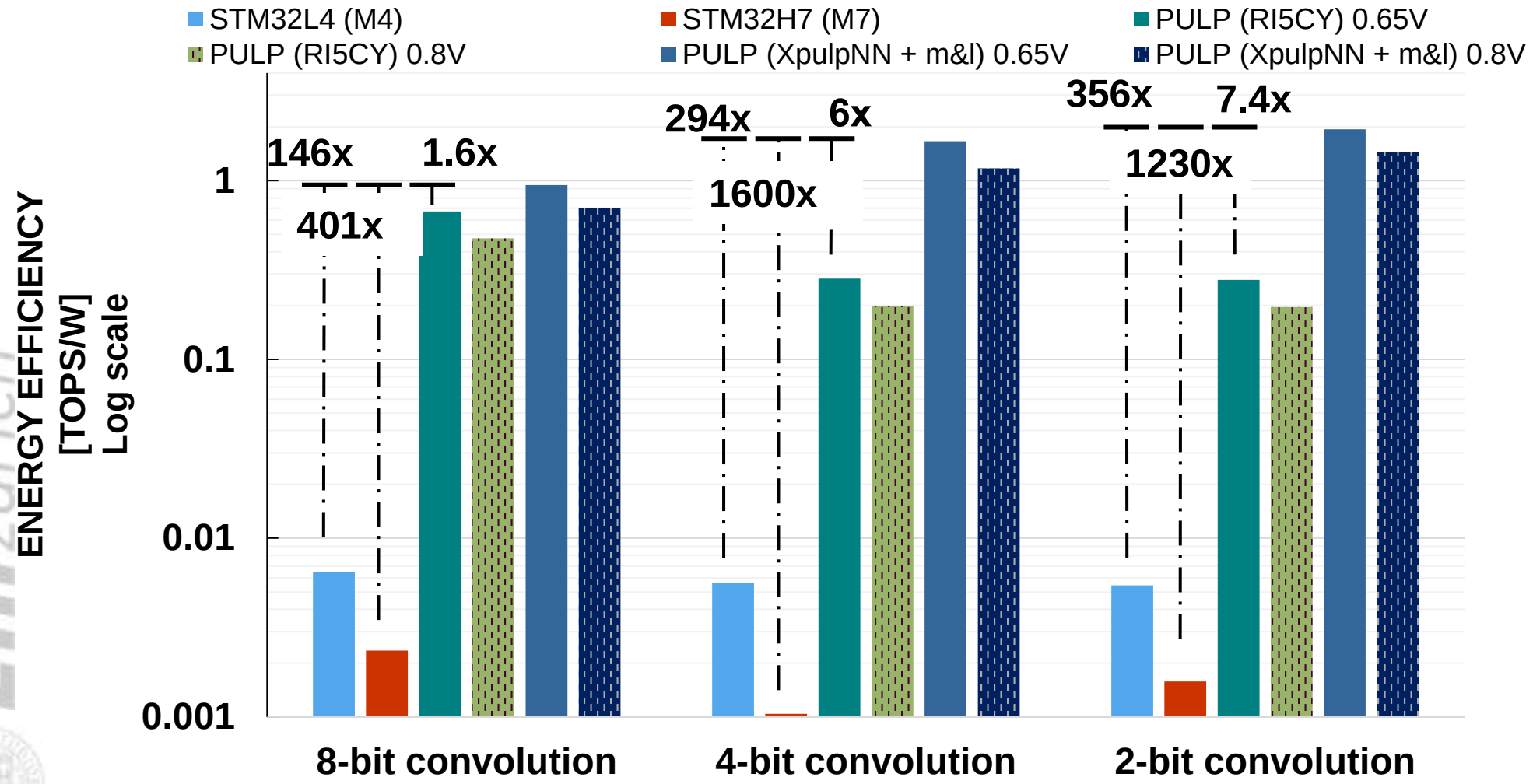
Virtual SIMD Instructions

- Encode operation as a virtual SIMD in the ISA (e.g. sdotsp.v)
- Format specified at runtime by a Control Register (e.g. 4x4)
- 180→18 Instructions needed for SIMD DOTP
- Potential to avoid code replica for different formats
- Tiny Overhead on QNN for Switching format
  - Format switch not frequent in DNN, e.g. every layer.

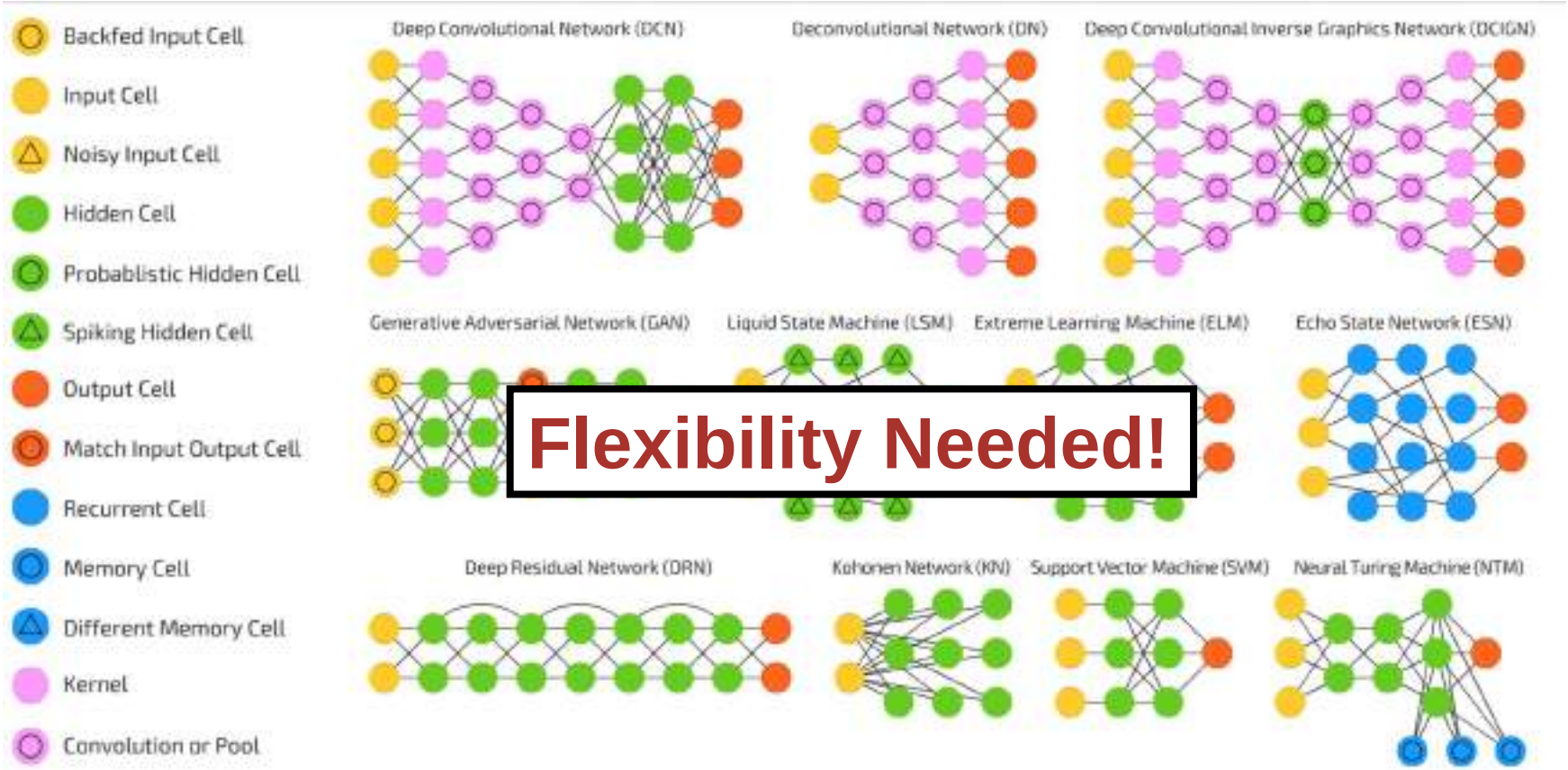




8-Cores Cluster + XpulpNN + M&L (22nm)



Recovering more efficiency? Sub-pJ/OP Accelerators

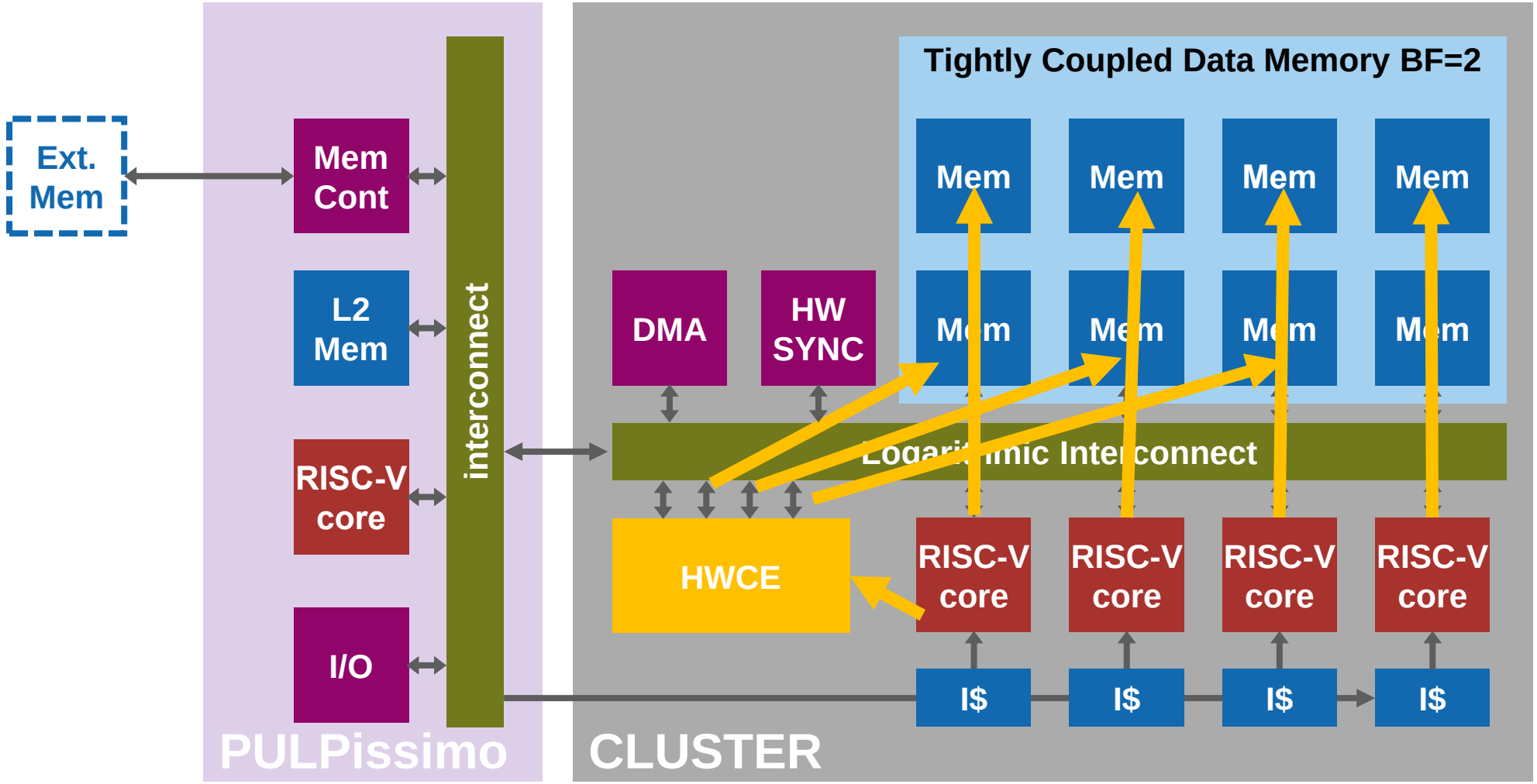


Flexibility Needed!

asimovinstitute.org/neural-network-zoo

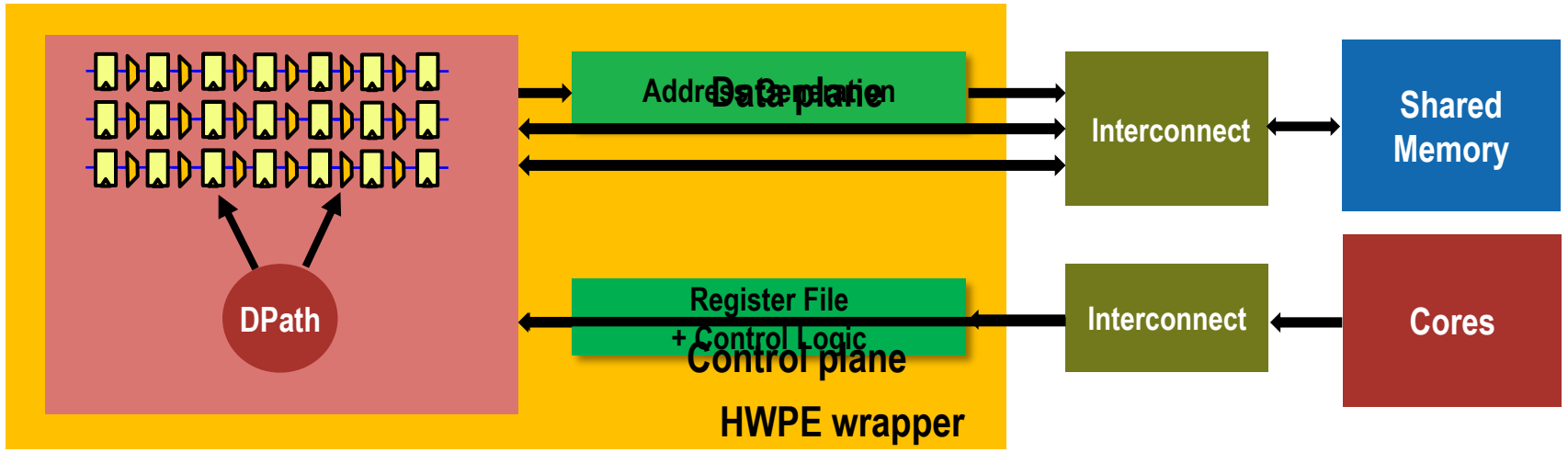
+ FFT, PCA, SVM, Mat-inv,…

Tightly-coupled HW Compute Engine



Acceleration with flexibility: zero-copy HW-SW cooperation

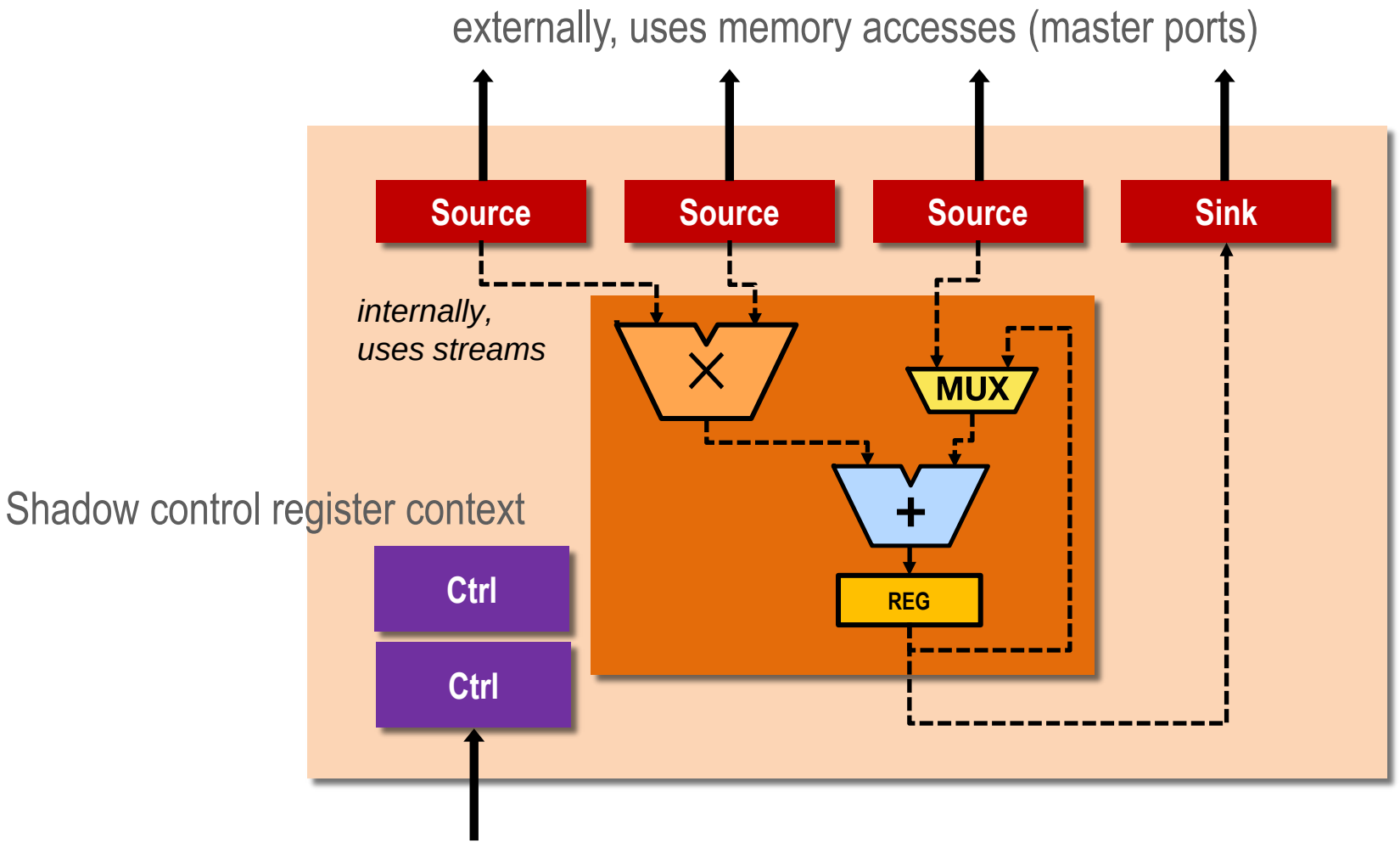
Hardware Processing Engines (HWPEs)



HWPE efficiency vs. optimized RISC-V core

1. Specialized datapath (e.g. systolic MAC) & internal storage (e.g. linebuffer, accum-regs)
2. Dedicated control (no I-fetch) with shadow registers (overlapped config-exec)
3. Specialized high-BW interco into L1 (on data-plane)

A toy HWPE

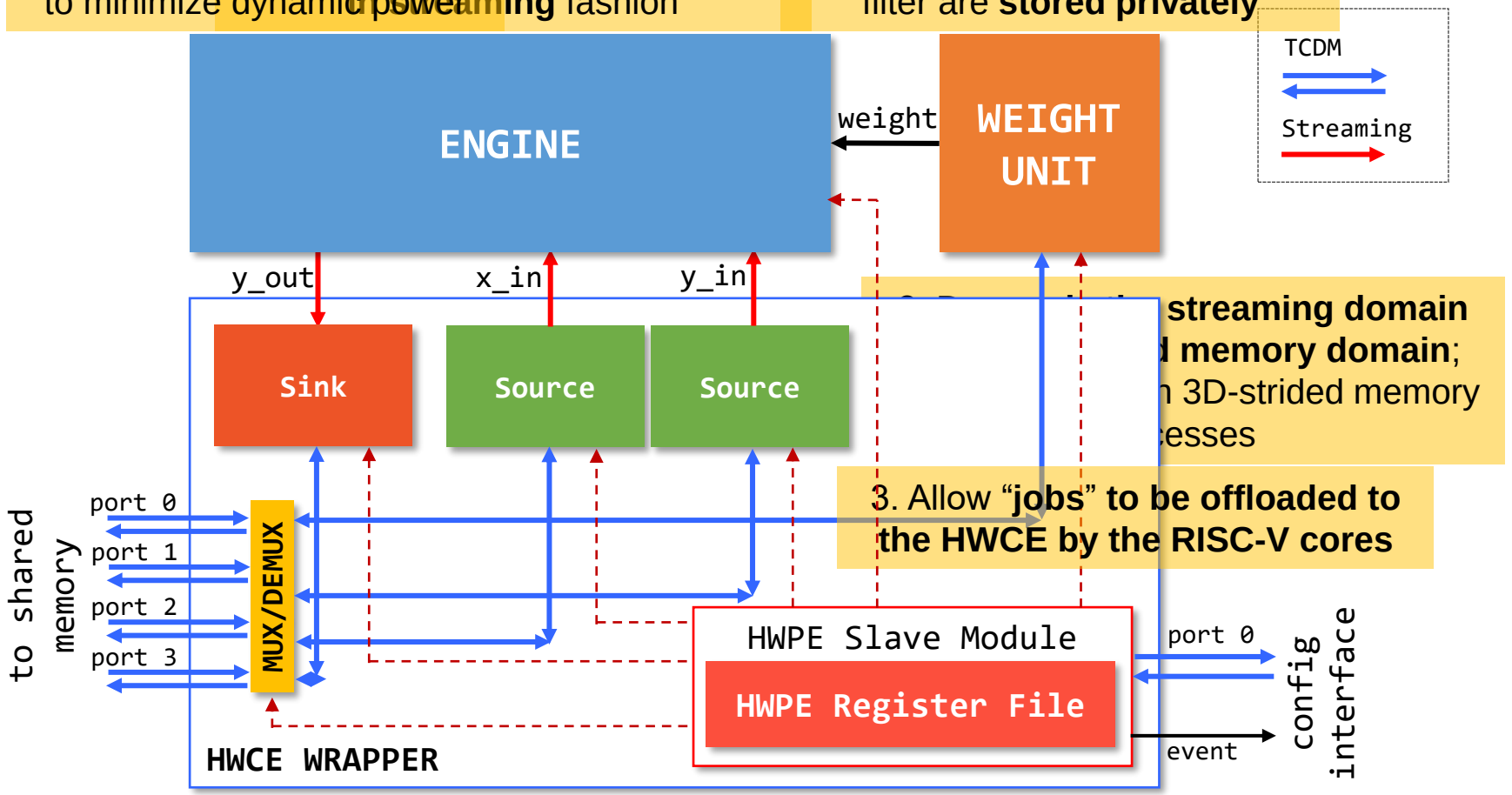

externally, uses memory accesses (master ports)
Source
Source
Source
Sink
internally,
uses streams
MUX
REG
Shadow control register context
Ctrl
Ctrl
Peripheral access (target port)

HW Convolution Engine

5. **Fine-grain clock gating** to minimize dynamic power

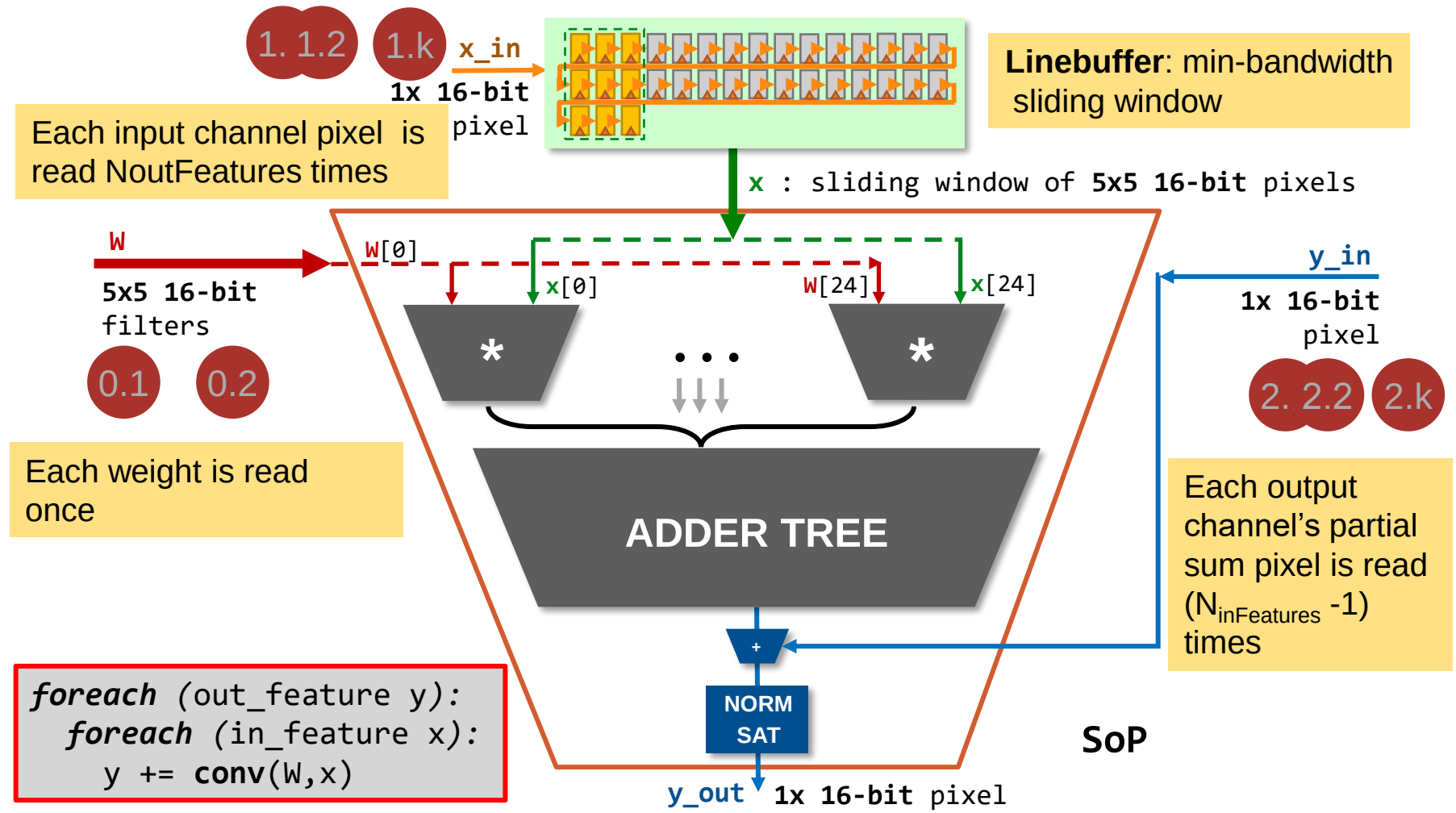
1. Configure convolve-accumulate **streaming** fashion

4. **Weights** for each convolution filter are **stored privately**



F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, pp. 683-688.

HWCE Sum-of-Products




```
foreach (out_feature y):
  foreach (in_feature x):
    y += conv(W,x)
```

Extreme Quantization→ Binarization with XNOR nets

$$\mathbf{y}(k_{out}) = \text{binarize}_{\pm 1}\left(\mathbf{b}_{k_{out}} + \sum_{k_{in}}\left(\mathbf{W}(k_{out}, k_{in}) \otimes \mathbf{x}(k_{in})\right)\right)$$

XNOR

$$\text{binarize}_{\pm 1}(t) = \text{sign}\left(\gamma\frac{t-\mu}{\sigma} + \beta\right)$$

$$\text{binarize}_{0,1}(t) = \begin{cases} 1 \text{ if } t \geq -\kappa/\lambda \doteq \tau, \text{ else } 0 & (\text{when } \lambda > 0) \\ 1 \text{ if } t \leq -\kappa/\lambda \doteq \tau, \text{ else } 0 & (\text{when } \lambda < 0) \end{cases}$$

Binary product → XOR

A	B	out
-1	-1	+1
-1	+1	-1
+1	-1	-1
+1	+1	+1

A	B	out
0	0	1
0	1	0
1	0	0
1	1	1

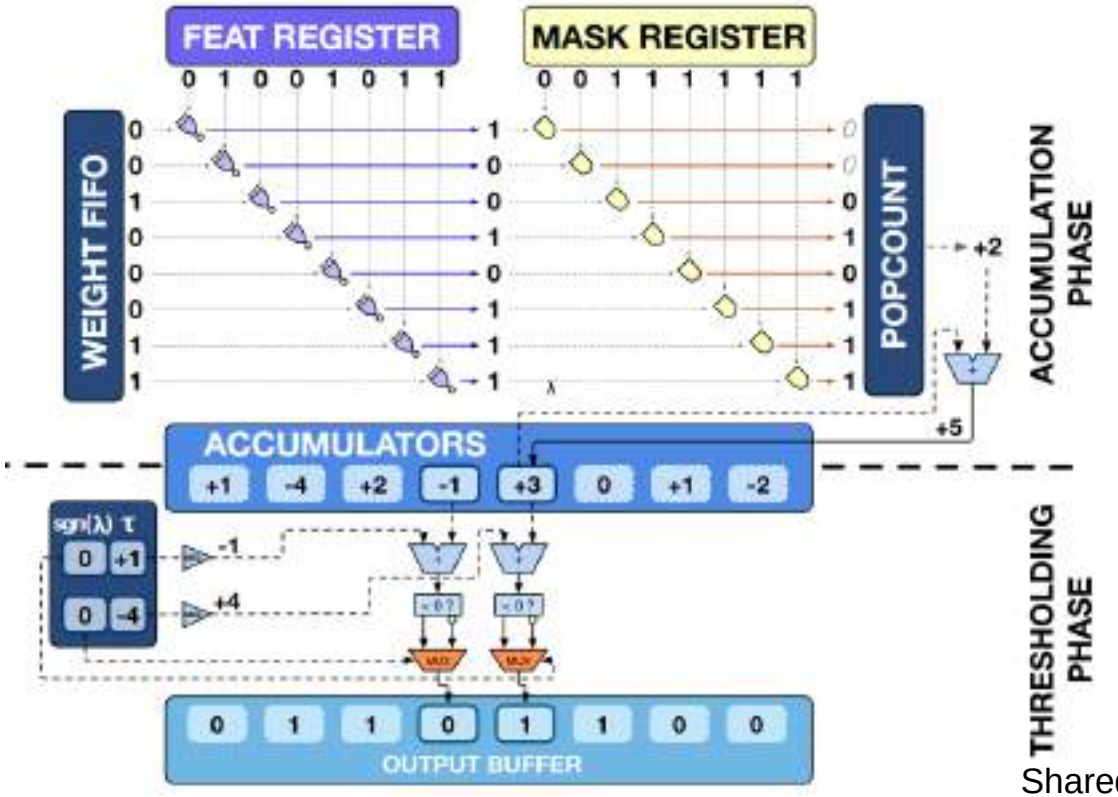
$$\mathbf{y}(k_{out}) = \text{binarize}_{0,1}\left(\sum_{k_{in}}\left(\mathbf{W}(k_{out}, k_{in}) \otimes \mathbf{x}(k_{in})\right)\right)$$

Thresholding

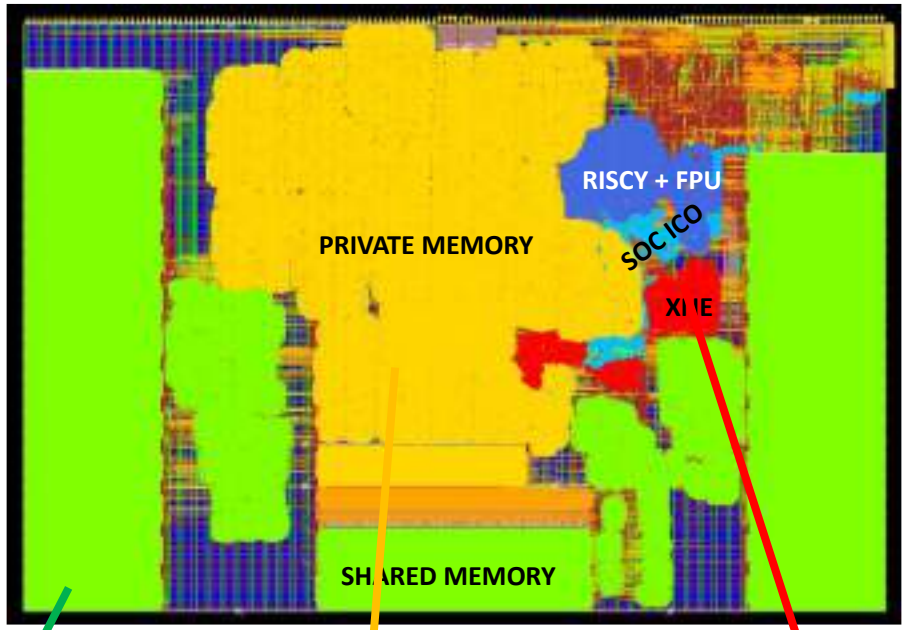
Multi-bit accumulation

XNE: XNOR Neural Engine

[Di Mauro et al. TCAS I, 2020]



Quentin in GlobalFoundries 22FDX



Shared memory is **56 KB** SRAM + **8 KB** SCM

Private memory is **448 KB** SRAM + 3r2w **8 KB** SCM

XNE area is **~14000 um2** (71 KGE, 72% Riscy+FPU)

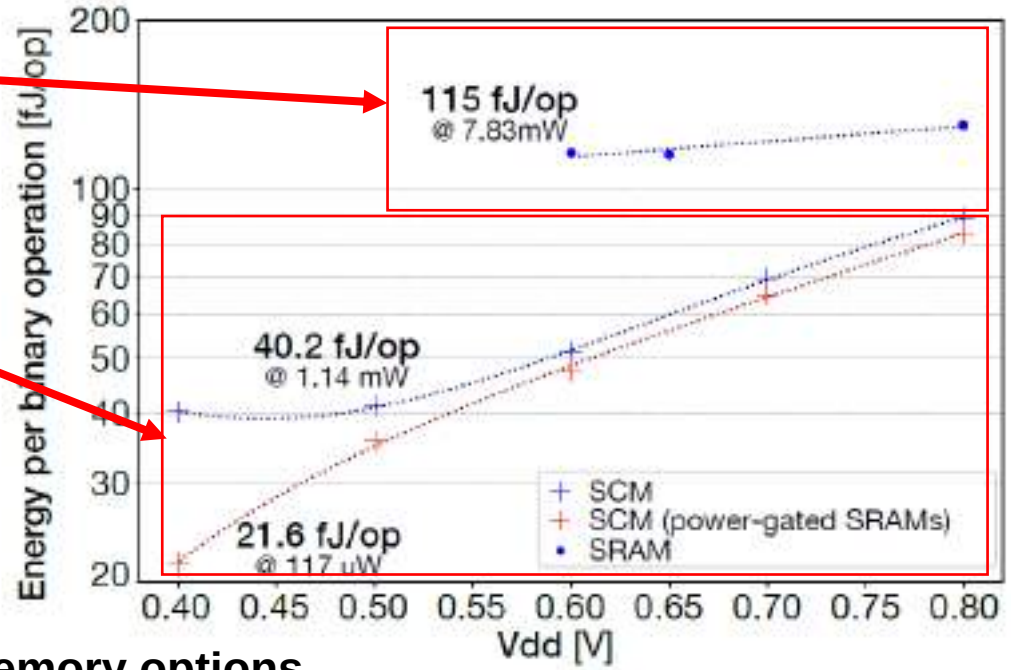
BINCONV: Binary dot-product and thresholding logic array

XNE Energy Efficiency

22 FDX measured silicon

With SRAMs, max eff
@ 0.65V **8.7 Top/s/W**

With SCMs, max eff
@ 0.5V **46.3 Top/s/W**



Note: All Memory on chip (max:MBs)

The importance of on-chip memory options
L1 SCM, L2 high-density, low leakgage SRAM (activations), MRAM (weights)

But… Accuracy Loss for BNNs is high even with retraining **(10%)**

Flexible allocation of precision is needed!

Flexibility needed: Binary-Based Quantization (BBQ)

QNN layer :

$$\mathbf{y}(k_{out}) = \boldsymbol{quant}\left(\sum_{k_{in}}\left(\mathbf{W}(k_{out}, k_{in}) \otimes \mathbf{x}(k_{in})\right)\right)$$

INT32 accumulator

Q-bit output fmaps

M-bit weights

N-bit input fmaps

Many $M \times N$ bits products...

... but one $M \times N$ product is the superposition of $M \times N$ **1-bit** products!

$$\mathbf{y}(k_{out}) = \boldsymbol{quant}\left(\sum_{i=0..M}\sum_{j=0..N}\sum_{k_{in}} 2^i 2^j \left(\mathbf{W_{bin}}(k_{out}, k_{in}) \otimes \mathbf{x_{bin}}(k_{in})\right)\right)$$

power-of-2 scaling factors

Q-bit output fmaps

1-bit weights

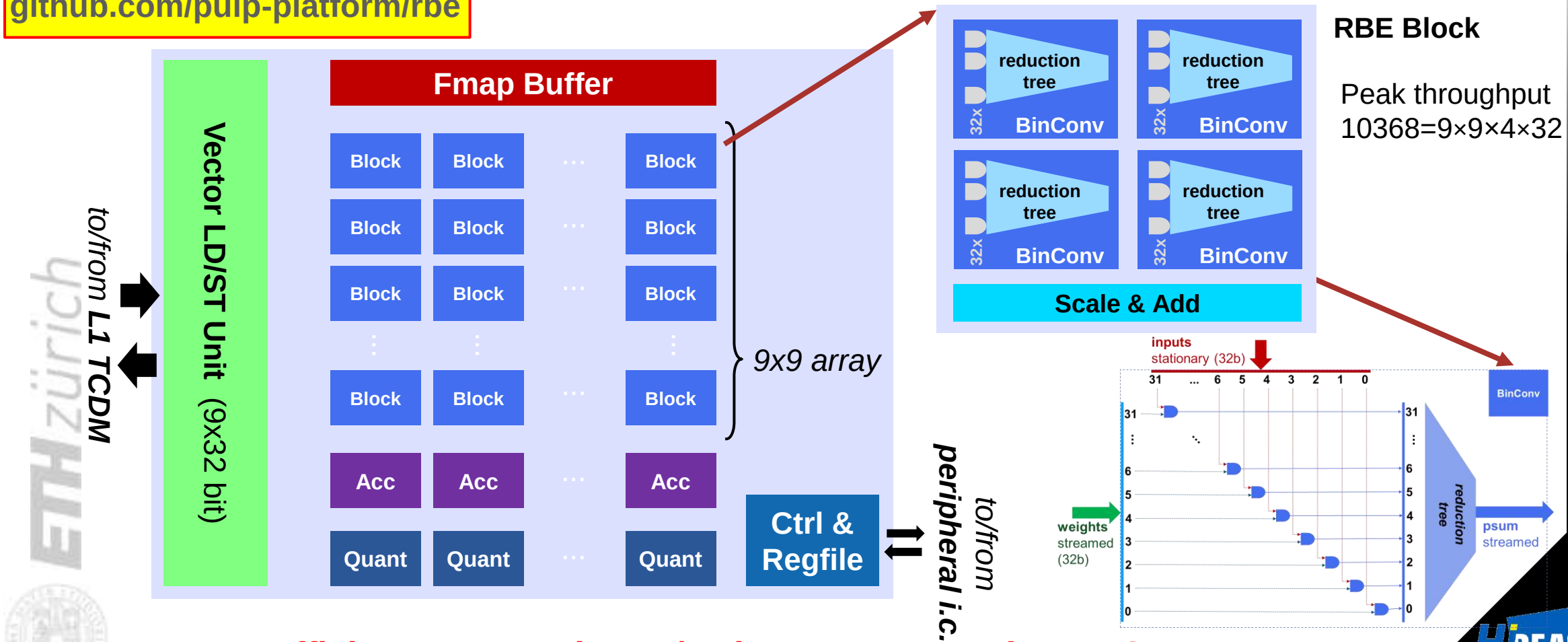
1-bit input fmaps

One quantized NN can be emulated by superposition of power-of-2 weighted $M \times N$ binary NN

Reconfigurable Binary Engine

$$\mathbf{y}(k_{out}) = \boldsymbol{quant}\left(\sum_{i=0..M}\sum_{j=0..N}\sum_{k_{in}} 2^i 2^j \left(\mathbf{W_{bin}}(k_{out}, k_{in}) \otimes \mathbf{x_{bin}}(k_{in})\right)\right)$$

github.com/pulp-platform/rbe

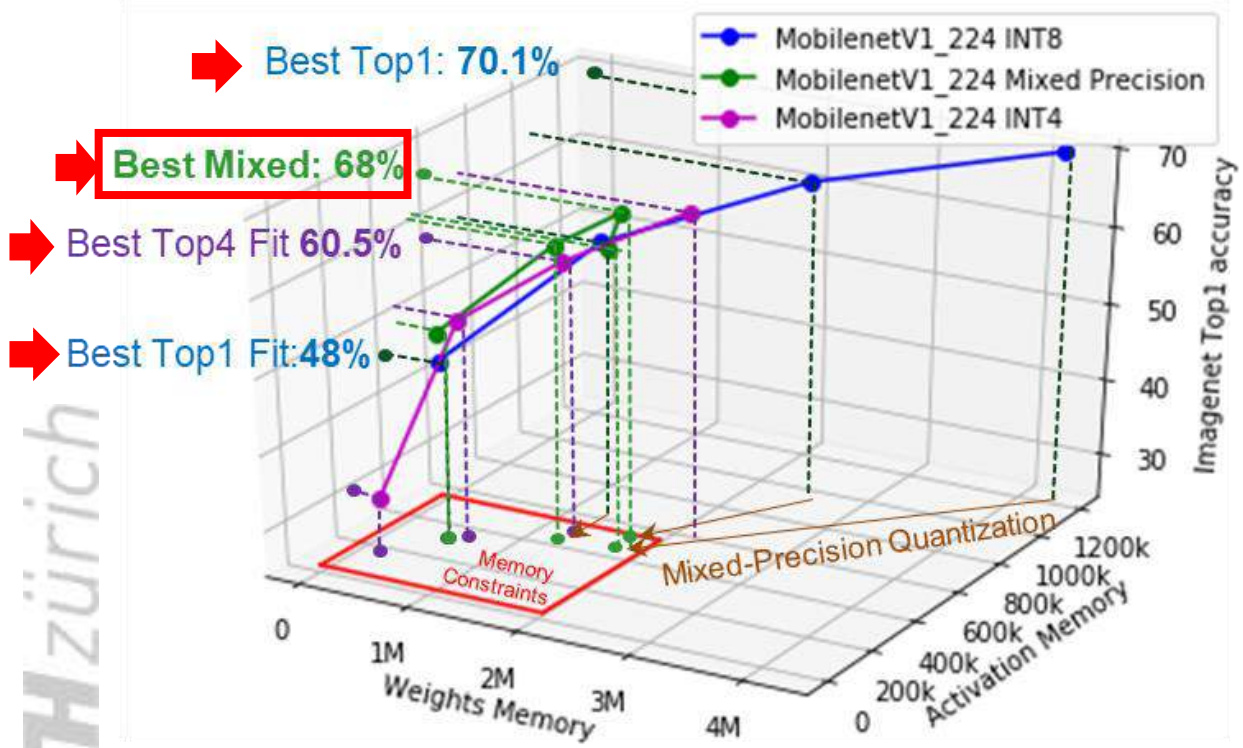


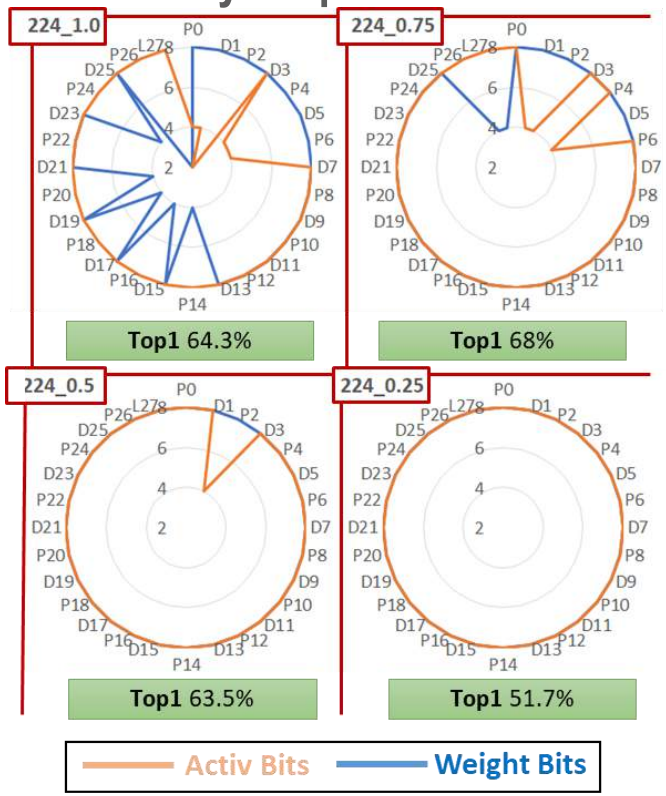
Energy efficiency 10-20x (0.1pJ/OP) wrt to SW on cluster @same accuracy

Mixed-Precision Quantized Networks – CMIX-NN

[Capotondi et al. TCAS II, 2020]

Apply tensor-wise quantization to fit memory constraints with low accuracy drop





Rule-based bit precision selection based on memory constraints

Avgbit 6.7 Act

Avgbit 5.9 Wgt

Only **-2%** wrt most accurate INT8 mobilenetV1 (224_1.0) which does not fit on-chip

+8% wrt most accurate INT8 mobilenetV1 fitting on-chip (192_0.5)

+7.5% wrt most accurate INT4 mobilenetV1 (224_1.0) fitting on chip

HW acceleration in perspective

Using 22FDX tech, NT@0.6V, High utilization, minimal IO & overhead

Energy-Efficient RV Core → **20pJ (8bit)**

↓

ISA-based 10-20x →**1-2pJ (8bit)** ➡ XPULPV2 &**V3**

↓

Configurable DP 10-20x → **50-100fJ (4bit)** ➡ HWCE, RBE, **NE**

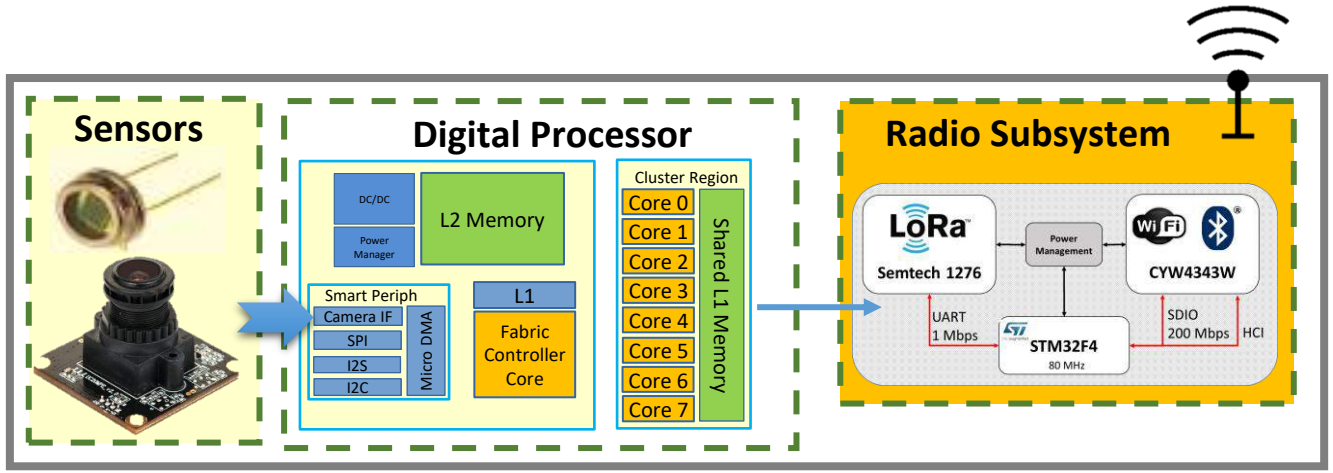
↓

Fully specialized DP 10-20x →**5-10fJ (ternary)** ➡ XNE, **CUTIE***

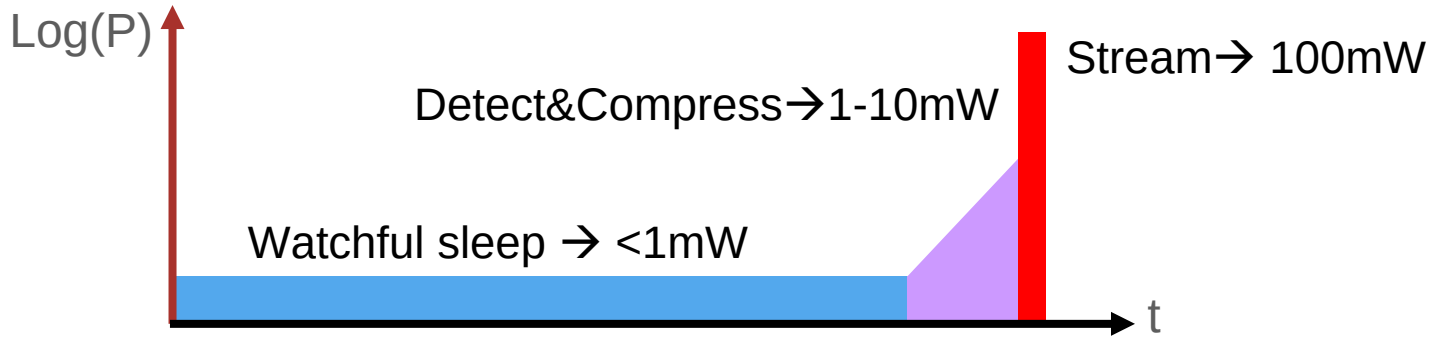
*See M. Scherer presentation, tinyML21 – sub 1fJ in 7nm

Towards In-Sensor: Achieving sub-mW average power?

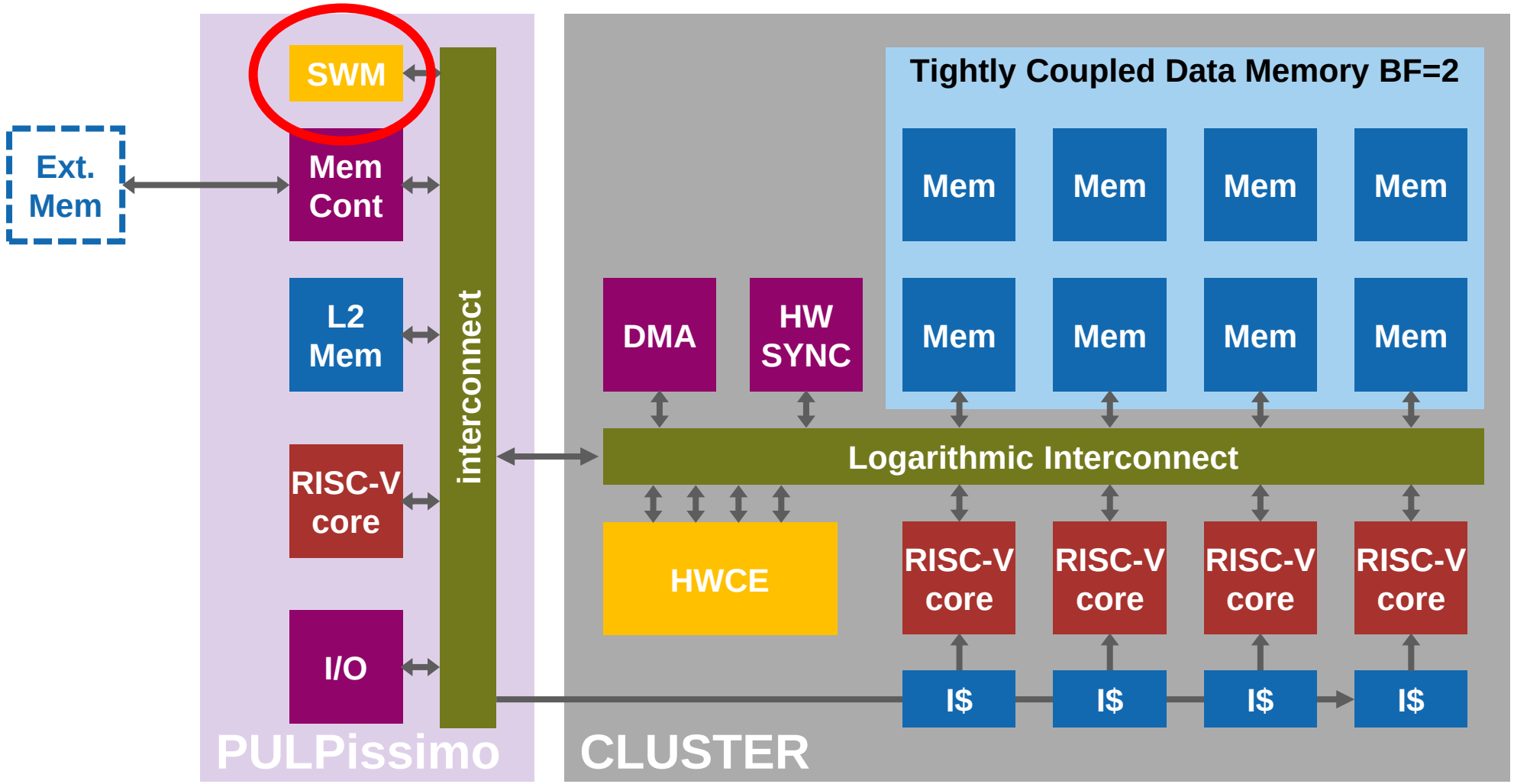
1mW average power with 10mW active power (10GOPS @ 1pJ/OP) → **sub mW sleep**



Duty cycling not acceptable when input events are asynchronous → **watchful Sleep**

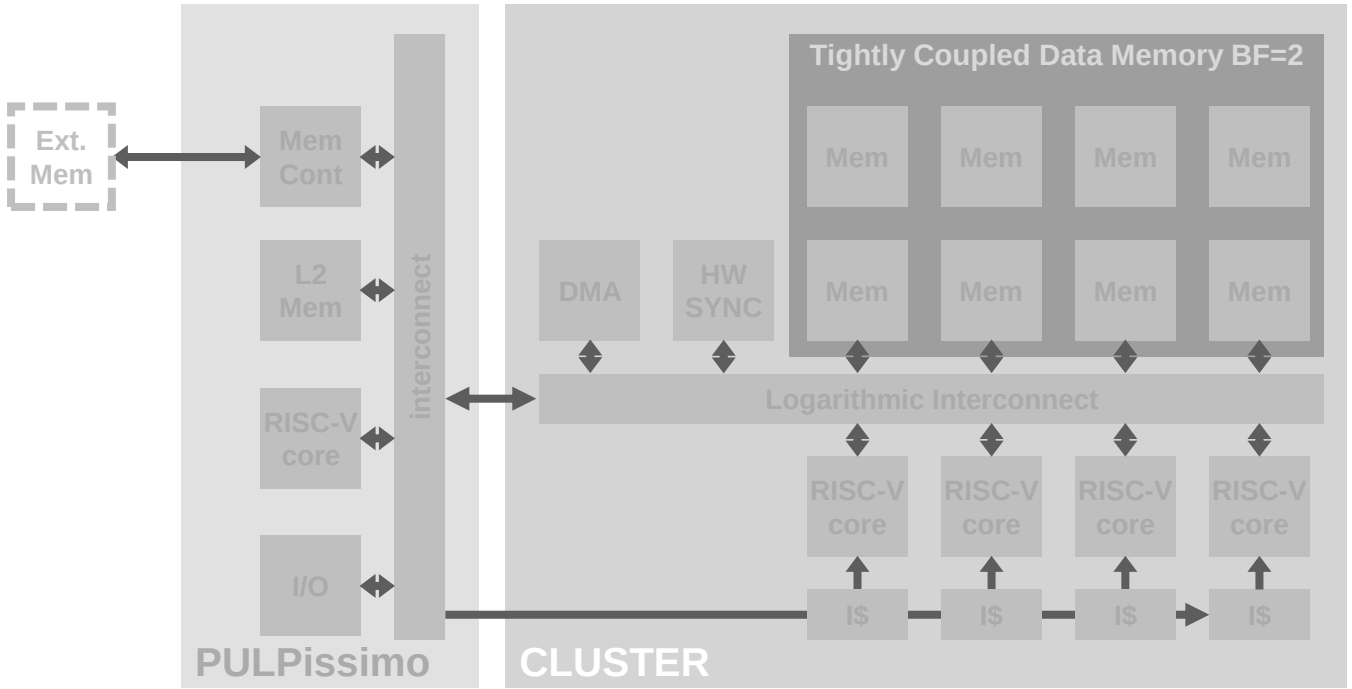


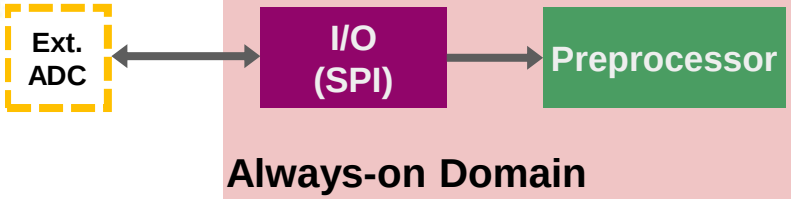
Need µW-range always-on Intelligence



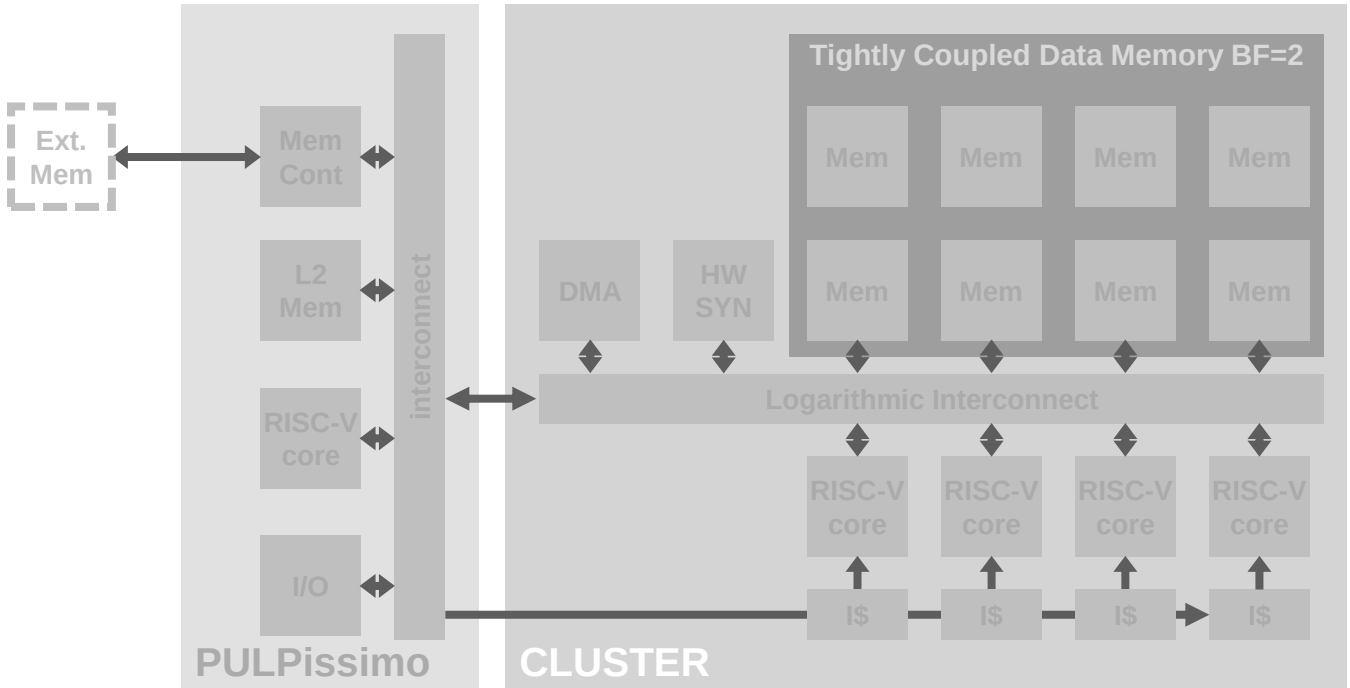
Smart Wakeup Module

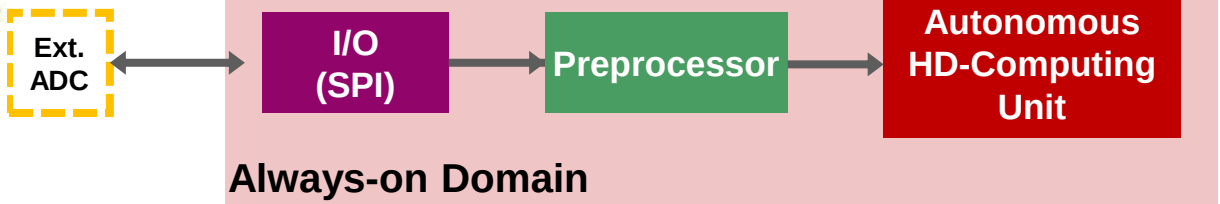
HD-Based smart Wake-Up Module



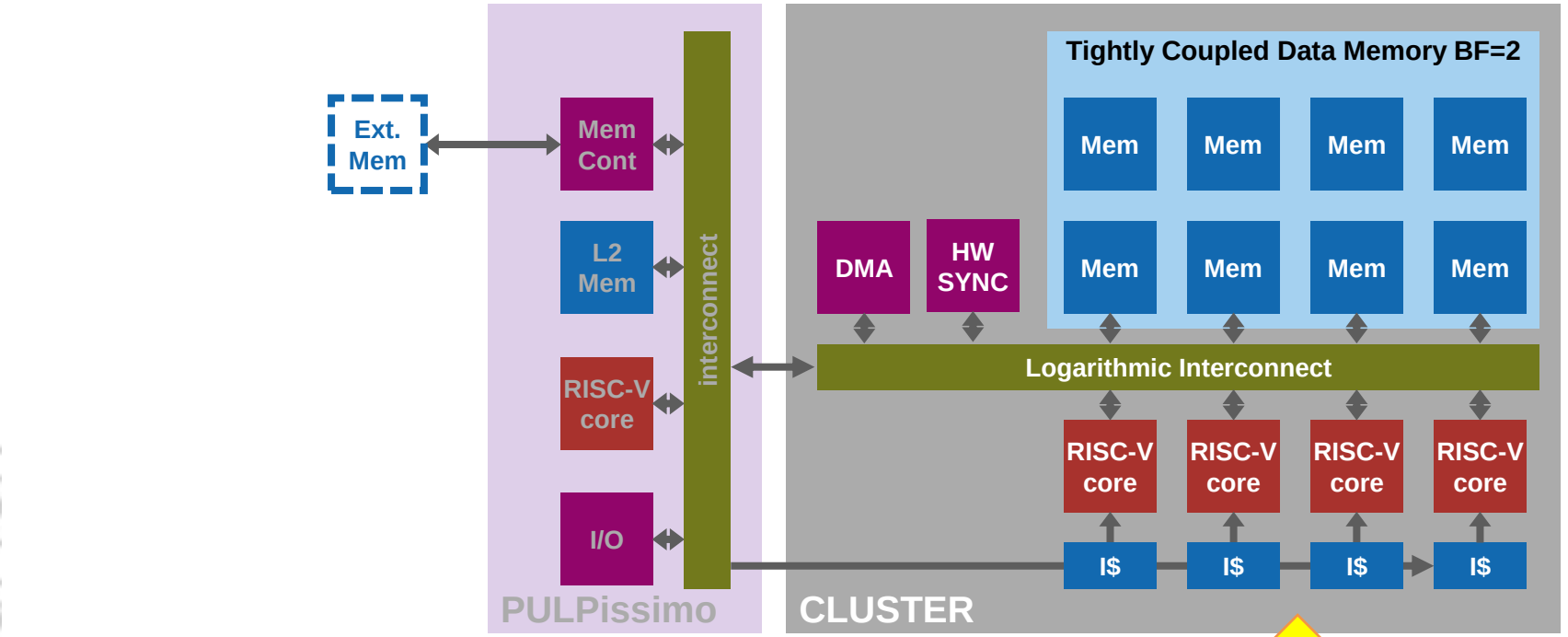


HD-Based smart Wake-Up Module

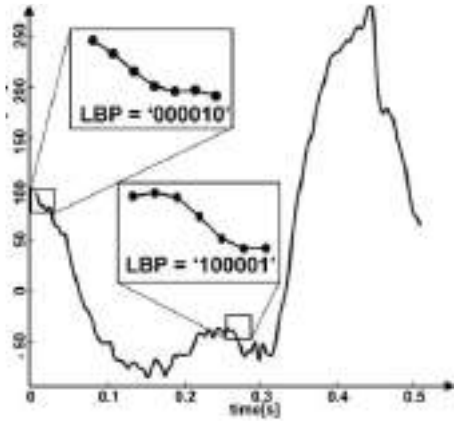




HD-Based smart Wake-Up Module



Not Only CNNs: Hyper-Dimensional Computing



Mapping →

1st 2nd 3rd 4th ... 1000th

[0 1 0 1 ..... 1]

Low Dimensional Input Data (e.g. 7-bit LBP)

[0 1 0 1 ..... 1]
[1 1 1 0 ..... 1]
[1 1 0 0 ..... 0]
[0 1 1 1 ..... 1]

HD-Encoding

- Component-wise Majority
- XOR
- Permutation

Search Vector

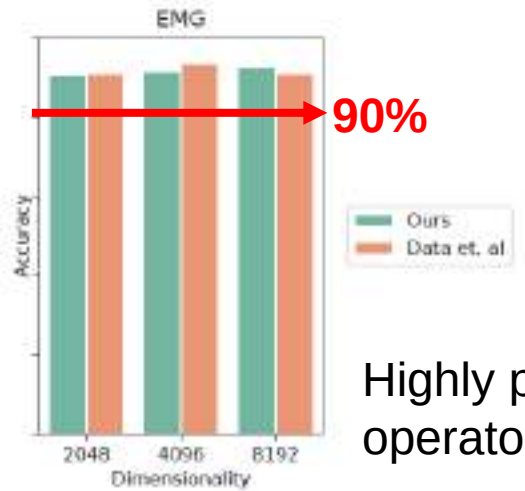
[1 1 0 1 ..... 1]

Similarity Search
(e.g. Hamming Distance)

Prototype Vectors

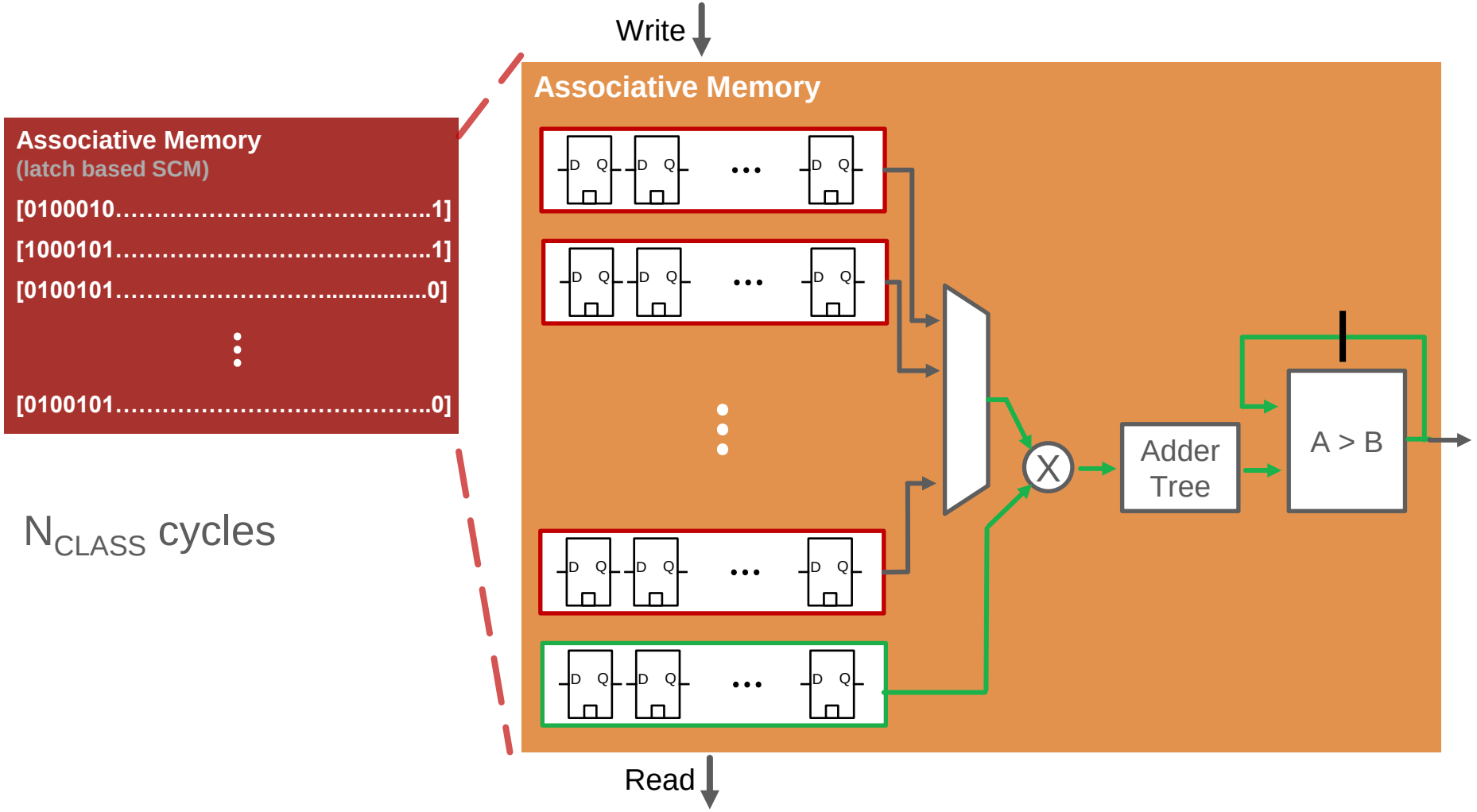
Associative Memory

[0 1 0 1 ..... 1]
[1 1 1 0 ..... 1]
[1 1 0 0 ..... 0]
[0 1 1 1 ..... 1]
[1 1 1 1 ..... 1]
[0 1 0 1 ..... 1]
[0 1 0 1 ..... 1]



Highly parallel, fault-tolerant binary operators, assoc-min-distance search → Merge storage & computation i.e. **In-memory computing**

In-memory Hyperdimensional Computing

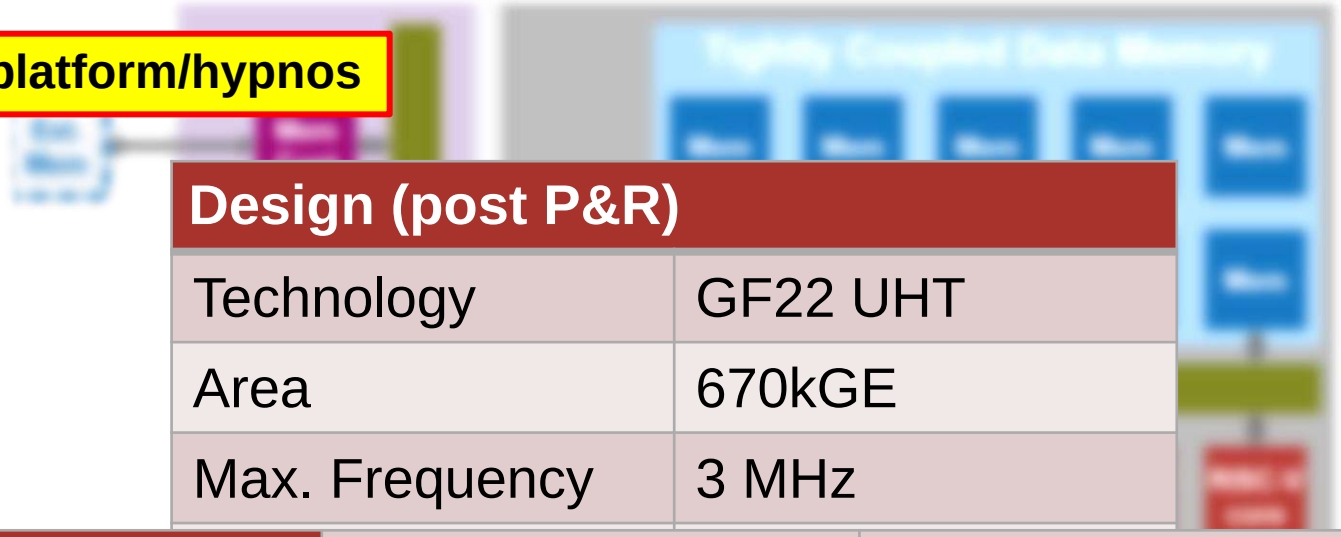


N_{CLASS} cycles

HD-Based smart Wake-Up Module - Hypnos

[Eggiman et al. arxiv.org/abs/2102.02758]

github.com/pulp-platform/hypnos



Design (post P&R)	
Technology	GF22 UHT
Area	670kGE
Max. Frequency	3 MHz

Implemented with lowest leakage cell library (UHVT)

f_{clk}	32kHz	200kHz
max. sampling rate	150 SPS/Channel	1kSPS/Channel
$P_{SWU,\ dynamic}$	0.99uW	6.21uW
$P_{SWU,\ leakage}$	0.7uW	0.7uW
$P_{SPI,\ dynamic}$	1.28uW	8.00uW
$P_{SWU,\ total}$ **Measured**	**2.97uW**	**14.9uW**

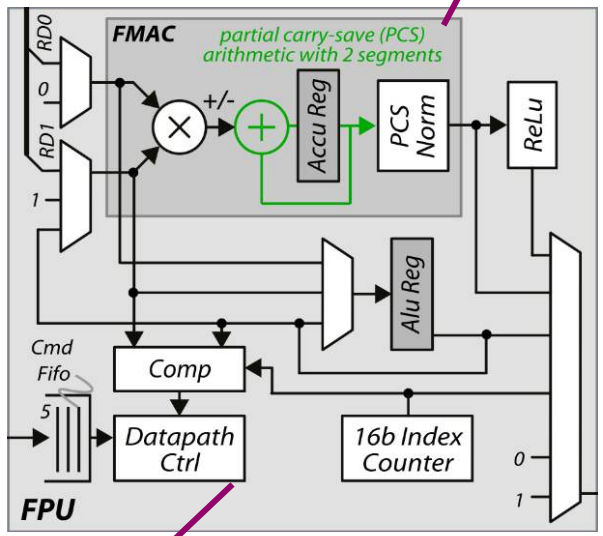
PULP in HPC?

- So far, focus on **low-power** applications: near-sensor processing, nano-UAVs, etc…
- ... what about **high-performance computing**?
  - energy efficiency of capital importance (**power = $$$** spent for **cooling**, **energy bill**)
- **Hardware accelerators** provide a key technology for HPC
  - compute-dominated workloads
  - highly parallel workloads
  - efficiency in Joules/op and power envelope in kW are important metrics
  - flexibility is also of primary importance
- Our focus so far has been on **artificial intelligence**
  - deep inference + deep learning: NTX

NTX: Boosting HWPEs for Deep Learning

The **Neural Training Accelerator** (**NTX**) [4] is built around a ***float32 fused multiply-accumulate*** core specialized for deep learning applications (ReLU, masking…)

32 bit inputs, ~300 bit partial carry-save accumulator, normalization upon write to memory.

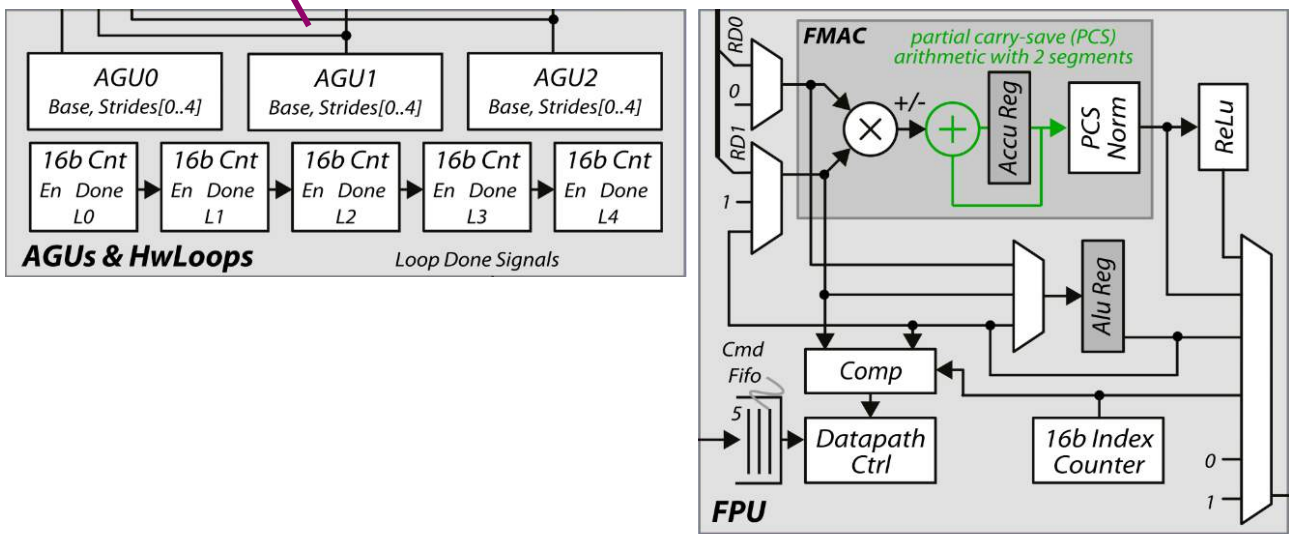


Datapath supports additional ReLU, comparison, and masking operations (for DNN layer derivatives)

NTX: Boosting HWPEs for Deep Learning

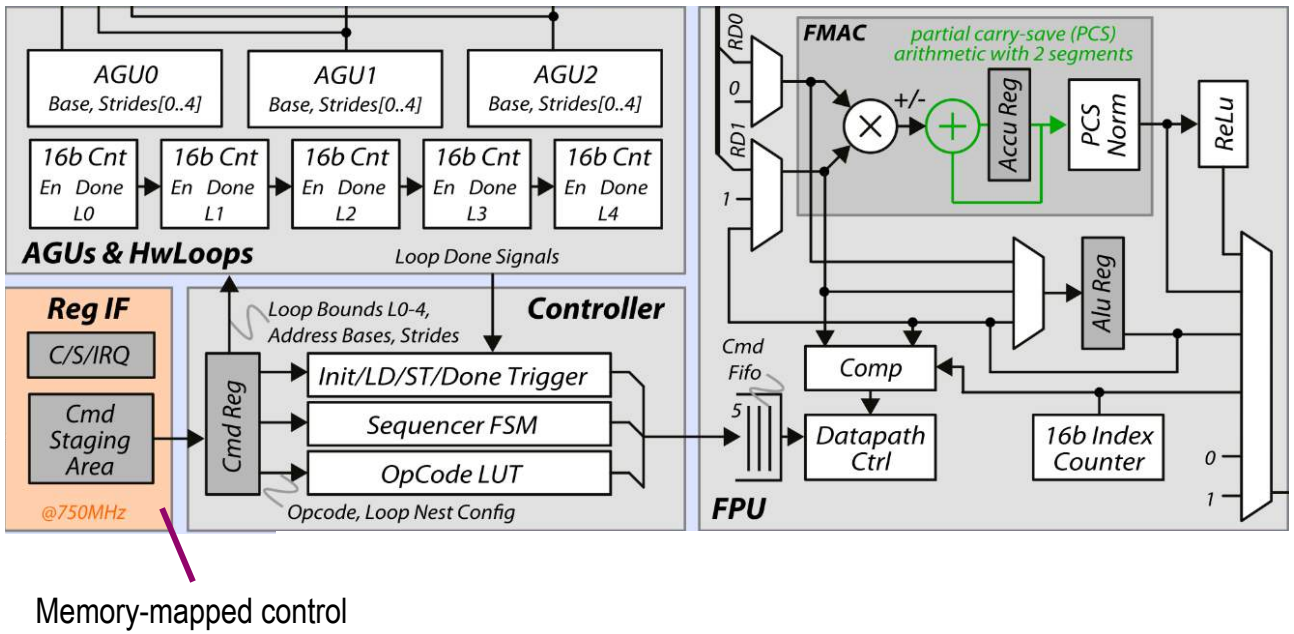
The **Neural Training Accelerator** (**NTX**) [4] is built around a ***float32 fused multiply-accumulate*** core specialized for deep learning applications (ReLU, masking…)

5 nested hardware loops and three address generators



NTX: Boosting HWPEs for Deep Learning

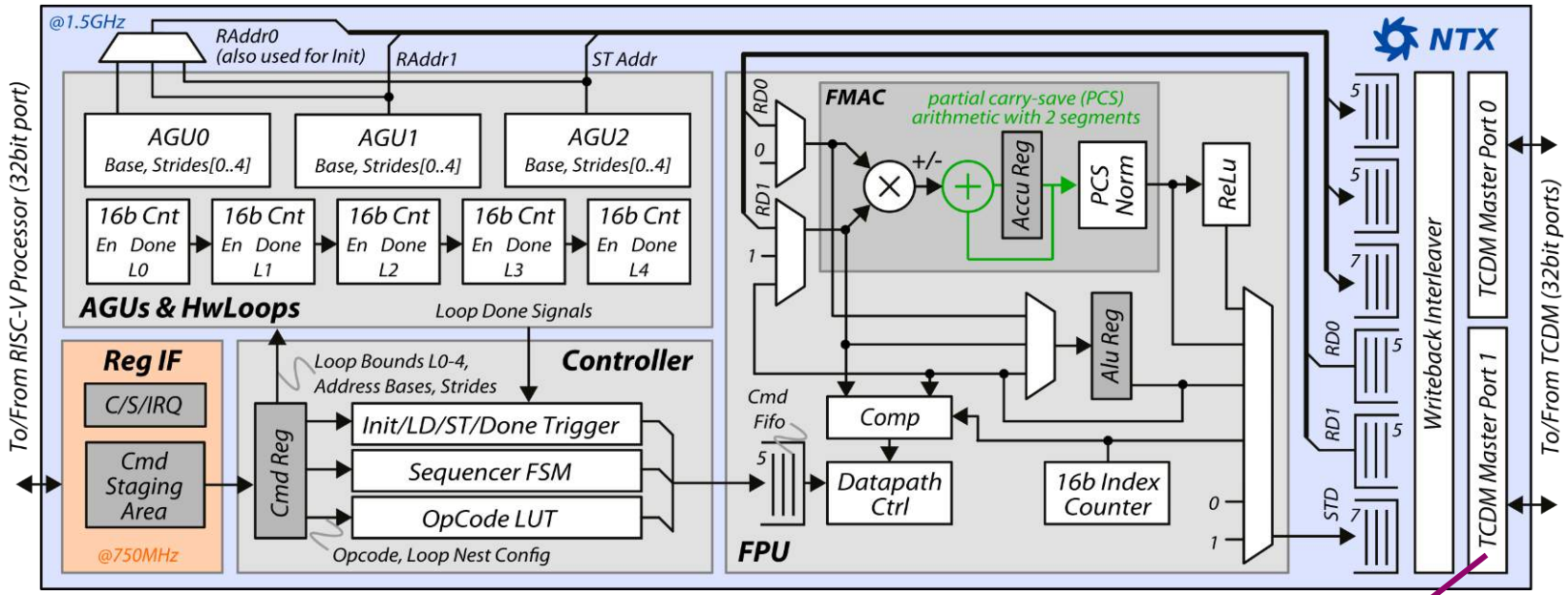
The **Neural Training Accelerator** (**NTX**) [4] is built around a ***float32 fused multiply-accumulate*** core specialized for deep learning applications (ReLU, masking…)



Memory-mapped control

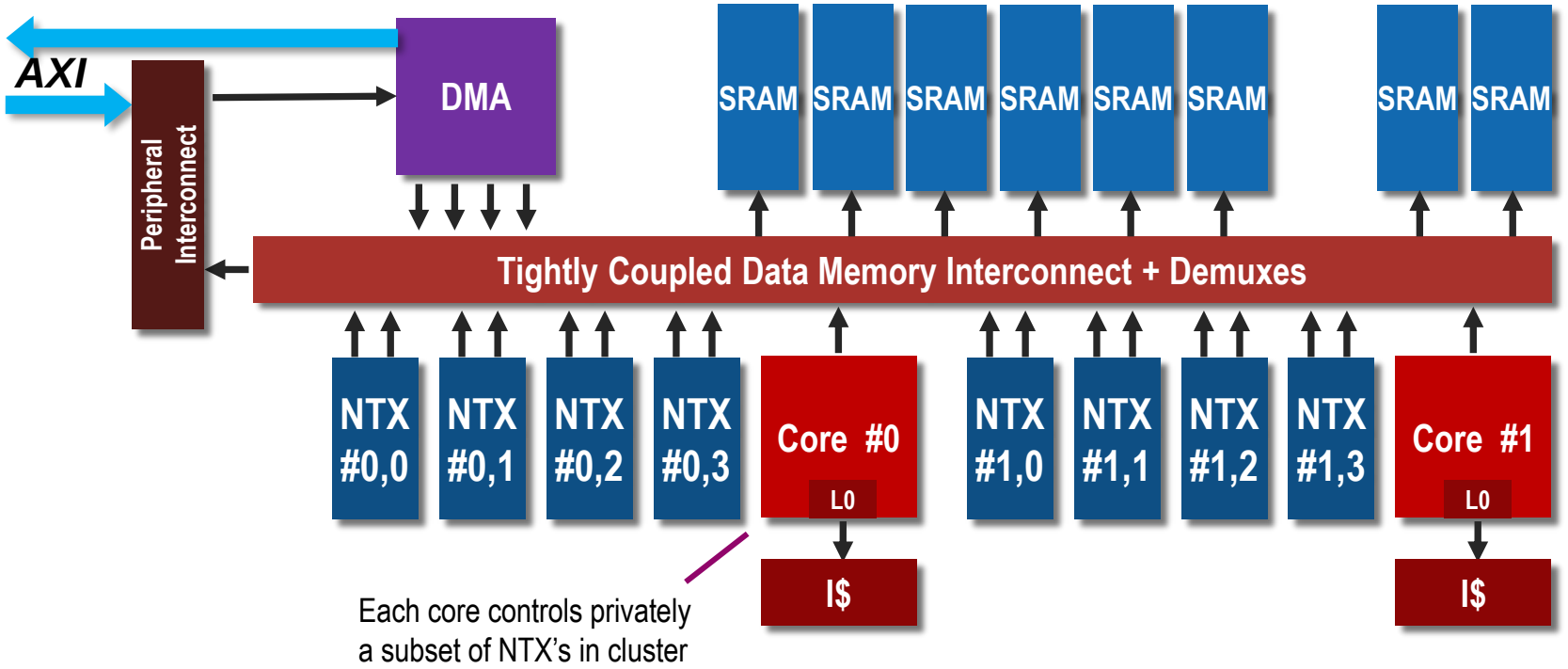
NTX: Boosting HWPEs for Deep Learning

The **Neural Training Accelerator** (**NTX**) [4] is built around a ***float32 fused multiply-accumulate*** core specialized for deep learning applications (ReLU, masking…)



2 ports into cluster memory read operands and write back results.

NTX-Augmented Clusters



Computation in a cluster is dominated by accelerators (*NTX*)

F. Schuiki, M. Schaffner, F. K. Gürkaynak and L. Benini, "A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets," in *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 484-497, 1 April 2019.

NST Stream MAC example

ONCE FOR EACH 4D TILE ⓐ

```
ISSUE_CMD(MEM_LDC, AGU0_S0, 1);
ISSUE_CMD(MEM_LDC, AGU0_S1, TCI*(Sx-1));
ISSUE_CMD(MEM_LDC, AGU0_S2, TCI*(TXI*Sy-Sx*KX+1));
```
AG0 SETUP

```
ISSUE_CMD(MEM_LDC, AGU1_S0, 1);
ISSUE_CMD(MEM_LDC, AGU1_S1, 0);
ISSUE_CMD(MEM_LDC, AGU1_S2, 0);
```
AG1 SETUP

```
ISSUE_CMD(NST0, MEM_LDC, HWL_E0, TCI);
ISSUE_CMD(NST0, MEM_LDC, HWL_E1, KX);
ISSUE_CMD(NST0, MEM_LDC, HWL_E2, KY);
```
HWLOOP SETUP

```
for (b=0; b<NUM_BATCHES; b++) {
```

AGs START ADDRESS SETUP

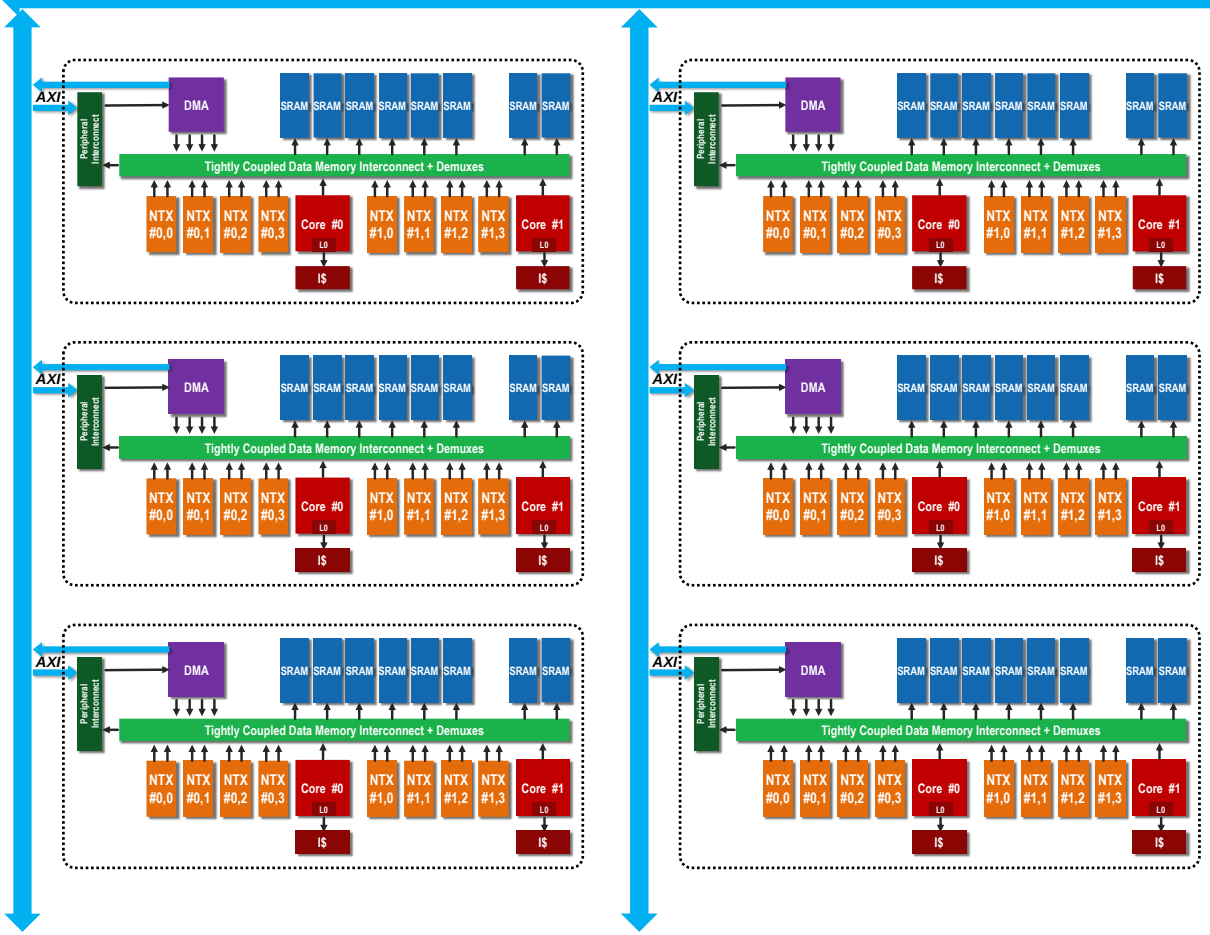
```
  ISSUE_CMD(NST0, MEM_LDC, AGU0_A, &kernels[tco][0][0][0]);
  ISSUE_CMD(NST0, MEM_LDC, AGU1_A, &tile_in[tyo][txo][0]);
  ISSUE_CMD(NST0, STREAM_MAC, 0, 0);
}
```
3D CONVOLUTION EXECUTION

ONCE FOR EACH 3D TILE

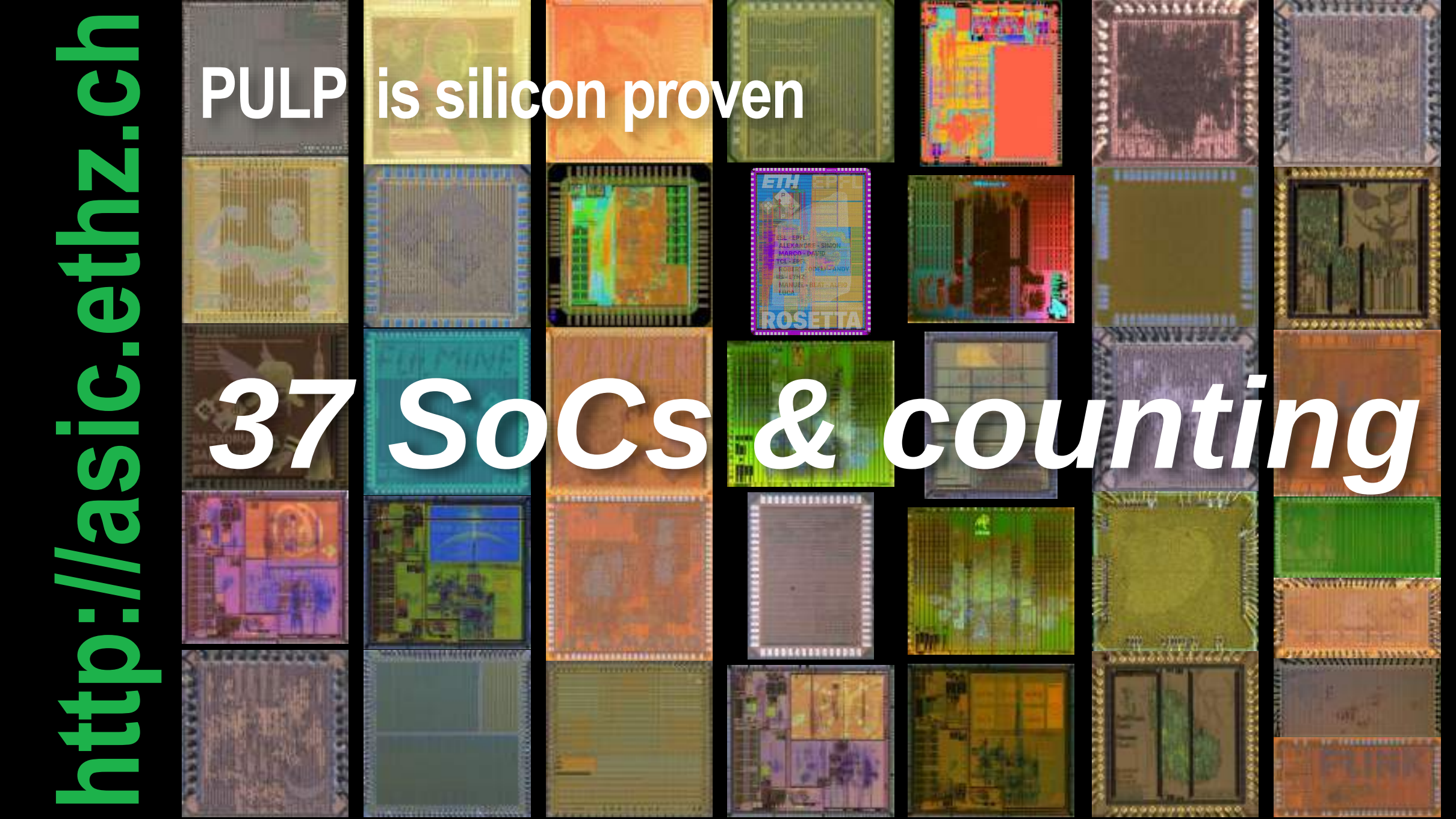
ⓑ

```
For k=0; k<E2; k++:
  For j=0; j<E1; j++:
    For i=0; i<E0; i++:
      ACC +=
        TCDM[AGU0] × TCDM[AGU1]
      AGU0 += S0_0
      AGU1 += S1_0
    AGU0 += S0_1
    AGU1 += S1_1
  AGU0 += S0_2
  AGU1 += S1_2
```

NTX-Augmented Clusters



Many clusters are connected through a higher level interconnect (e.g. AXI) to scale up computing performance


http://asic.ethz.ch
PULP is silicon proven
37 SoCs & counting
ETH
ROSETTA

PULP

Parallel Ultra Low Power

Luca Benini, Davide Rossi, Andrea Borghesi, Michele Magno, Simone Benatti, Francesco Conti, Francesco Beneventi, Daniele Palossi, Giuseppe Tagliavini, Antonio Pullini, Germain Haugou, Manuele Rusci, Florian Glaser, Fabio Montagna, Bjoern Forsberg, Pasquale Davide Schiavone, Alfio Di Mauro, Victor Javier Kartsch Morinigo, Tommaso Polonelli, Fabian Schuiki, Stefan Mach, Andreas Kurth, Florian Zaruba, Manuel Eggimann, Philipp Mayer, Marco Guermandi, Xiaying Wang, Michael Hersche, Robert Balas, Antonio Mastrandrea, Matheus Cavalcante, Angelo Garofalo, Alessio Burrello, Gianna Paulin, Georg Rutishauser, Andrea Cossettini, Luca Bertaccini, Maxim Mattheeuws, Samuel Riedel, Sergei Vostrikov, Vlad Niculescu, Hanna Mueller, Matteo Perotti, Nils Wistoff, Luca Bertaccini, Thorir Ingulfsson, Thomas Benz, Paul Scheffler, Alessio Burello, Moritz Scherer, Matteo Spallanzani, Andrea Bartolini, Frank K. Gurkaynak,
and many more that we forgot to mention

http://pulp-platform.org **@pulp_platform**