Hardware Acceleration in PULP

HPCA 2018 – Vienna (Austria)

25.2.2018

***Francesco Conti*[1,2]**
f.conti@unibo.it

ETH zürich
[1]*Integrated Systems Laboratory*

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
[2]*Department of Electrical, Electronic and Information Engineering*


Multitherman

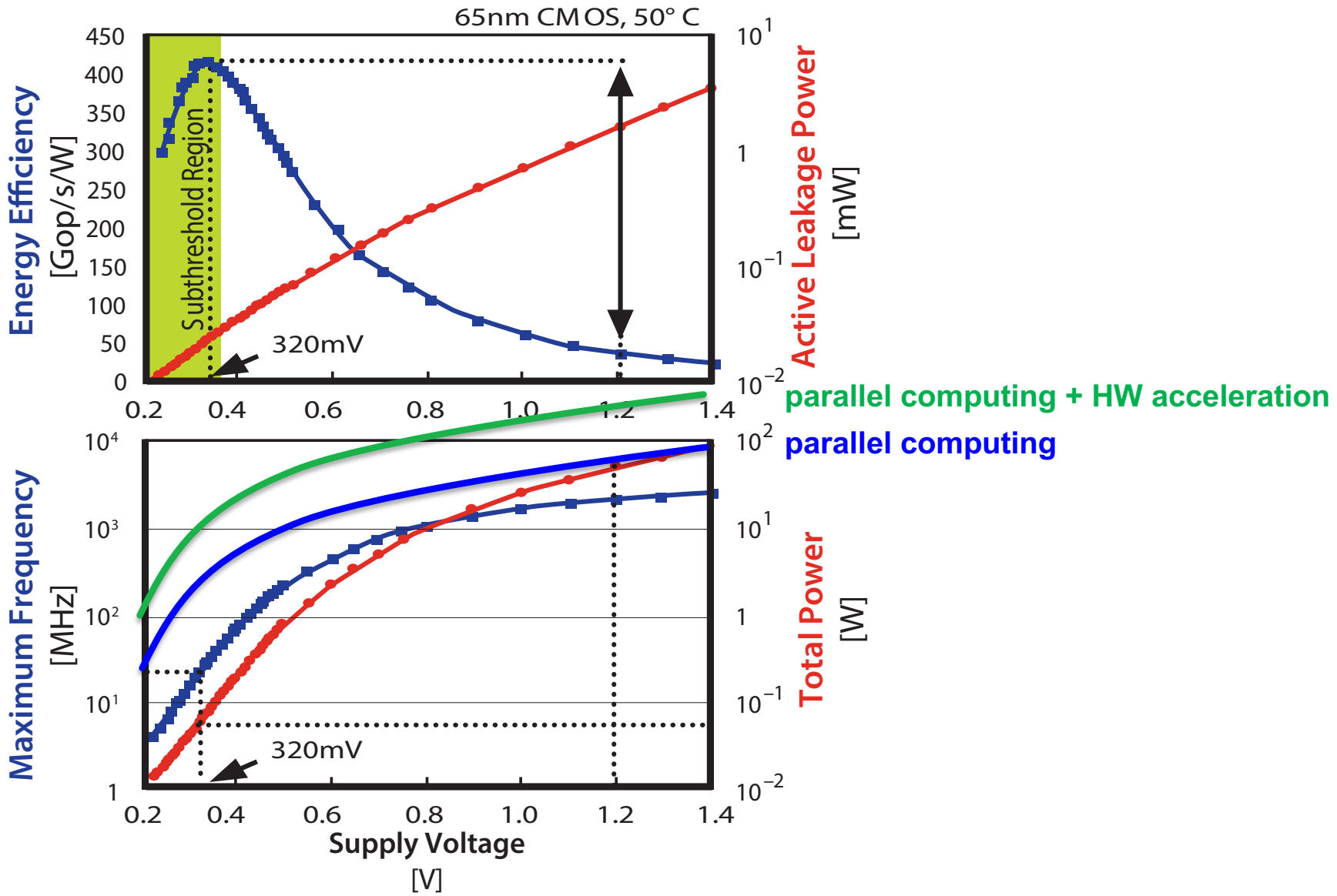
Why Hardware Acceleration, really?

- **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 **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
  - Accelerator guys also have to eat their daily pasta (important to this speaker)

Hardware Accelerators from a PULP-y Perspective

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



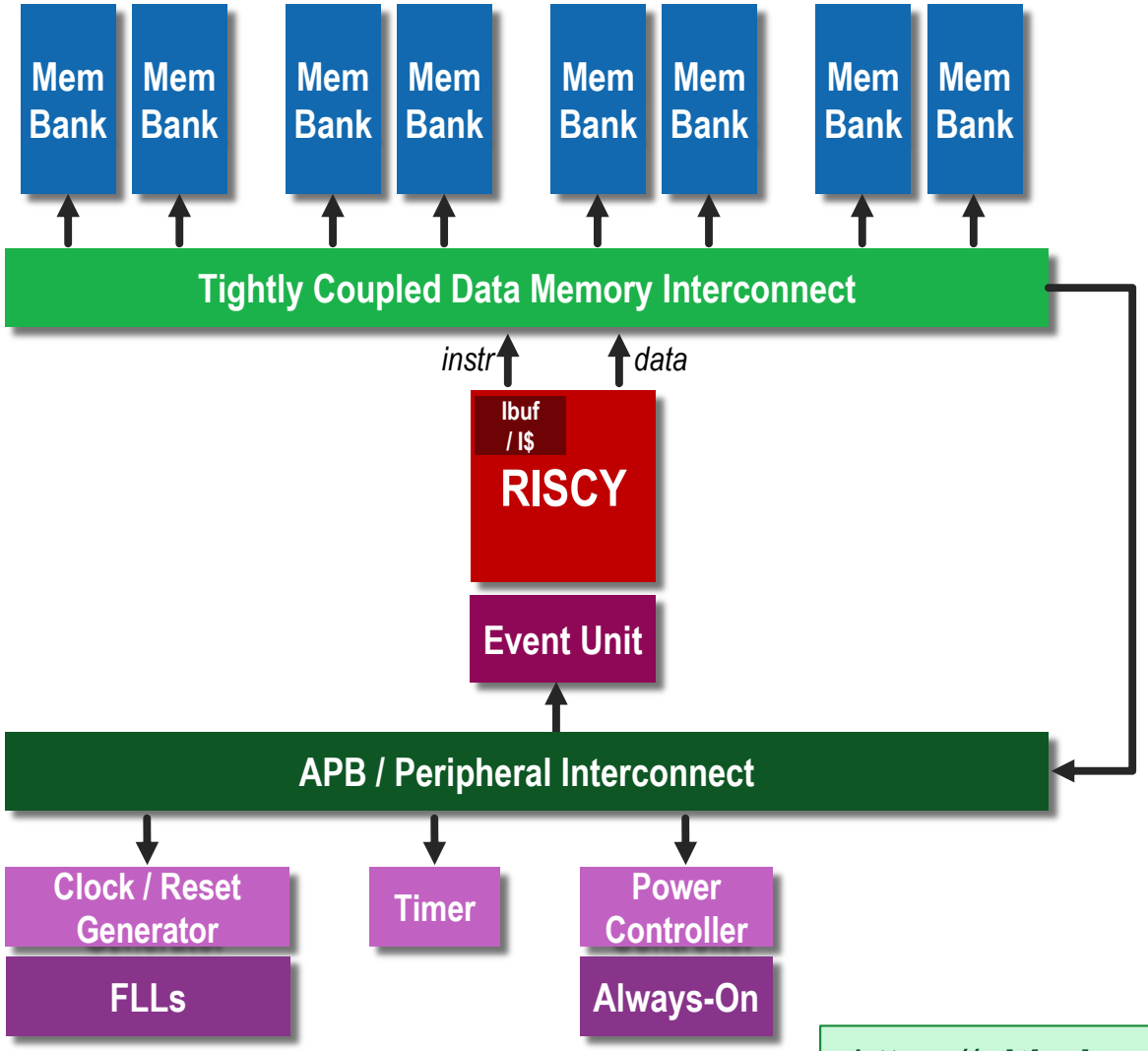
Hardware Accelerators from a PULP-y Perspective (2)

- **PULP Software**
  - which means that our **hardware accelerators** have to be SW-friendly
  - here we focus on relatively coarse-grained accelerators
    - not FPUs, not ASIPs, etc. → these require different considerations

- Accelerators must be **controllable** in a **SW-friendly** fashion
  - **memory-mapped** control like any other peripheral in the system (DMAs, I/O)
  - **regular LD/ST** for control: no special instructions, which require compiler changes (with little benefit)

- Accelerators must **exchange data** efficiently
  - **no communication through core register file** (very inefficient for coarse-grain)
  - **no need to use general platform DMAs** (specialized patterns are ill-supported, and it might be required for other jobs)

Hardware Accelerators from a PULP-y Perspective (3)

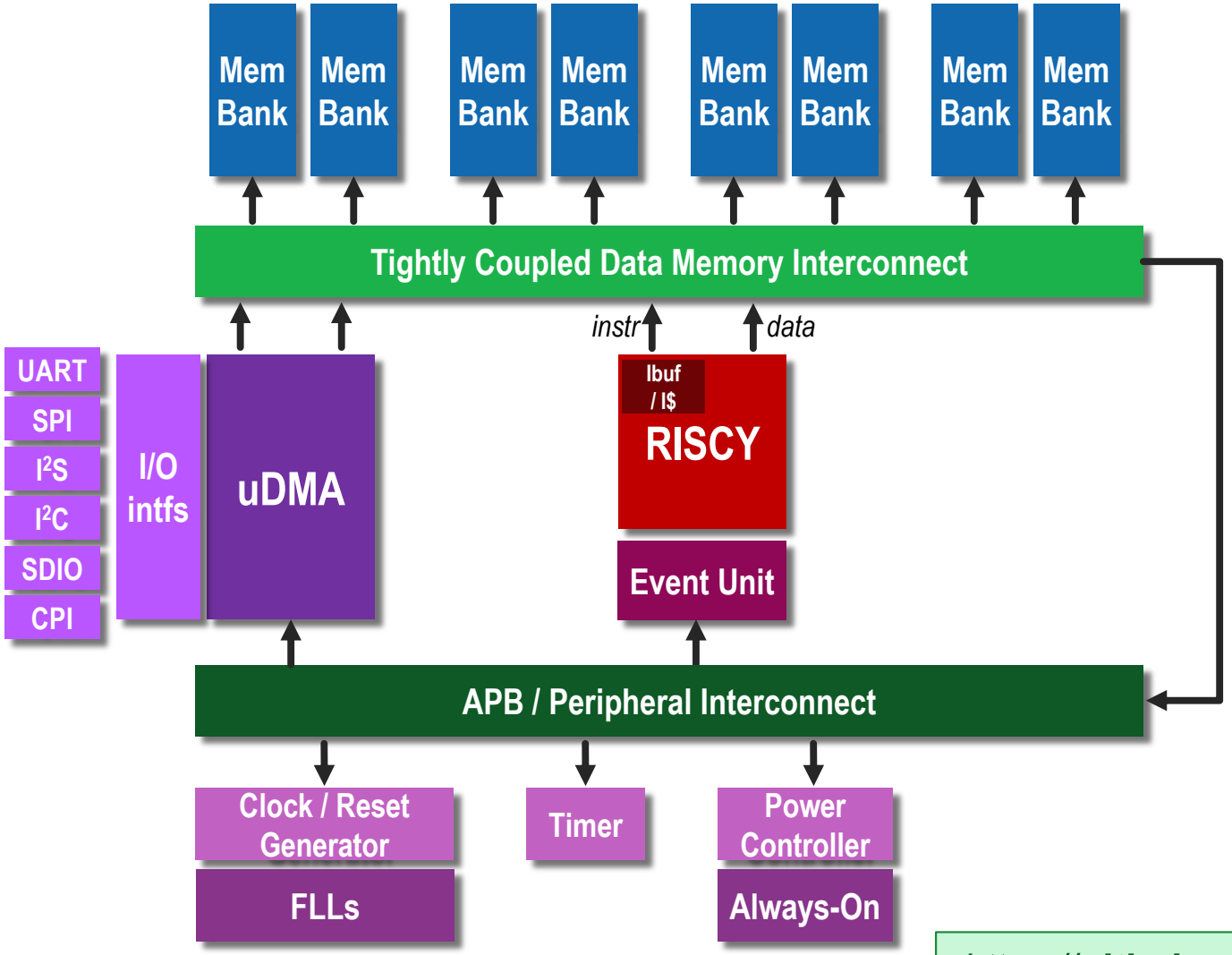


This is **PULPissimo**, our newest open-source microcontroller platform

1. Built on a **RISCY** or **Zero-RISCY** core

*https://**github.com/pulp-platform/pulpissimo***

Hardware Accelerators from a PULP-y Perspective (3)

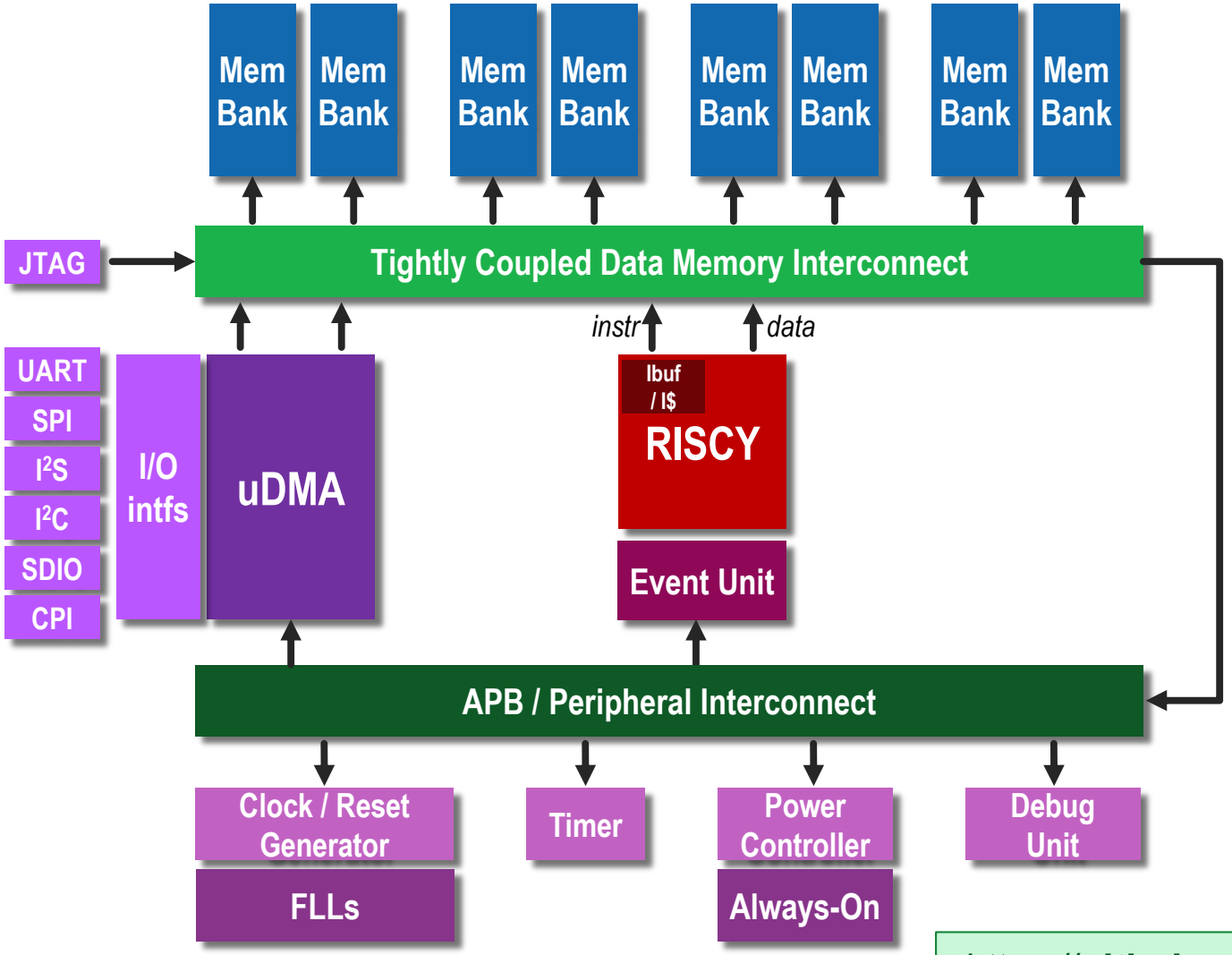


This is **PULPissimo**, our newest open-source microcontroller platform

1. Built on a **RISCY** or **Zero-RISCY** core
2. Featuring **uDMA** advanced shared-mem I/O subsystem

*https://**github.com/pulp-platform/pulpissimo***

Hardware Accelerators from a PULP-y Perspective (3)

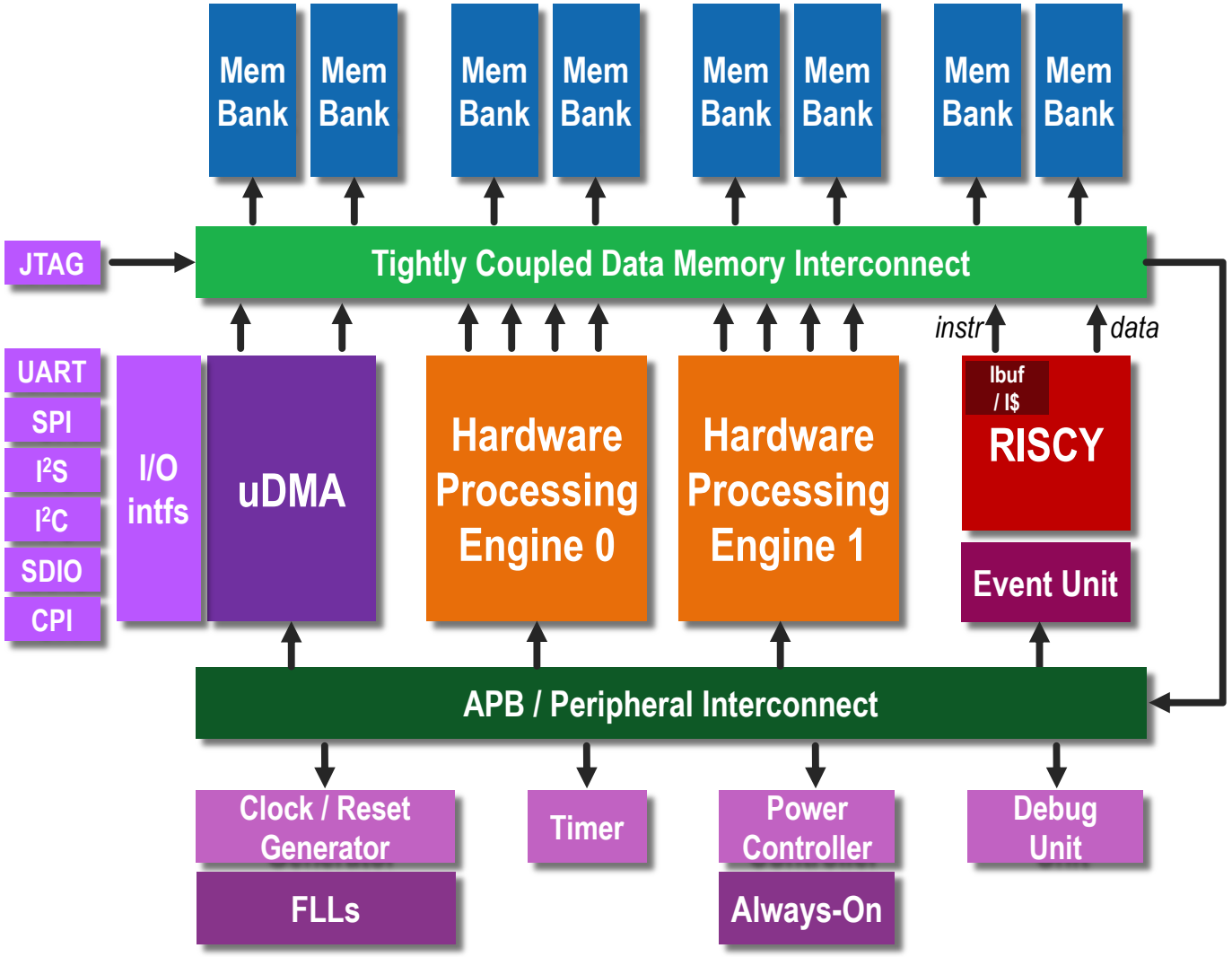


This is **PULPissimo**, our newest open-source microcontroller platform

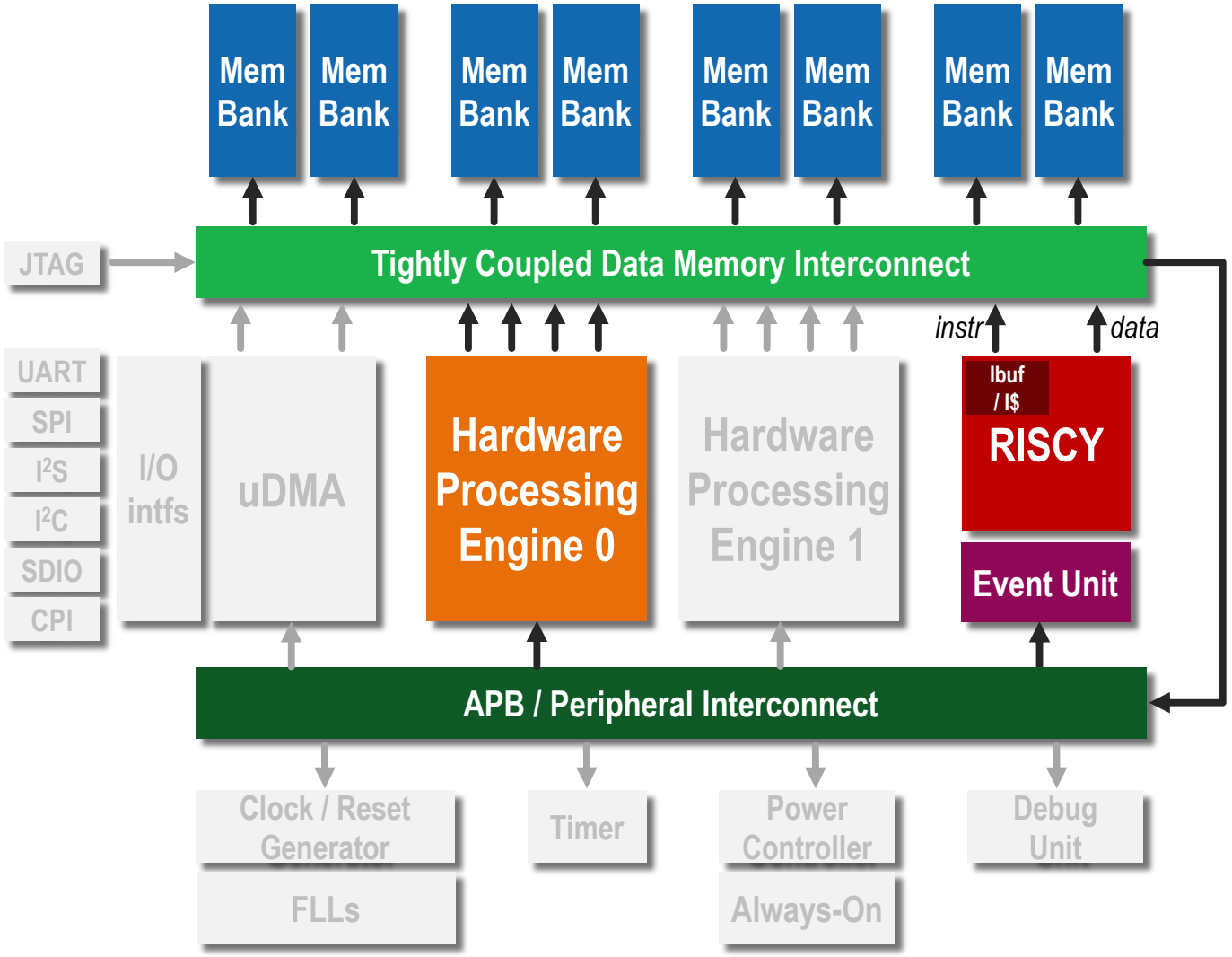
1. Built on a **RISCY** or **Zero-RISCY** core
2. Featuring **uDMA** advanced shared-mem I/O subsystem
3. JTAG-debuggable

*https://**github.com/pulp-platform/pulpissimo***

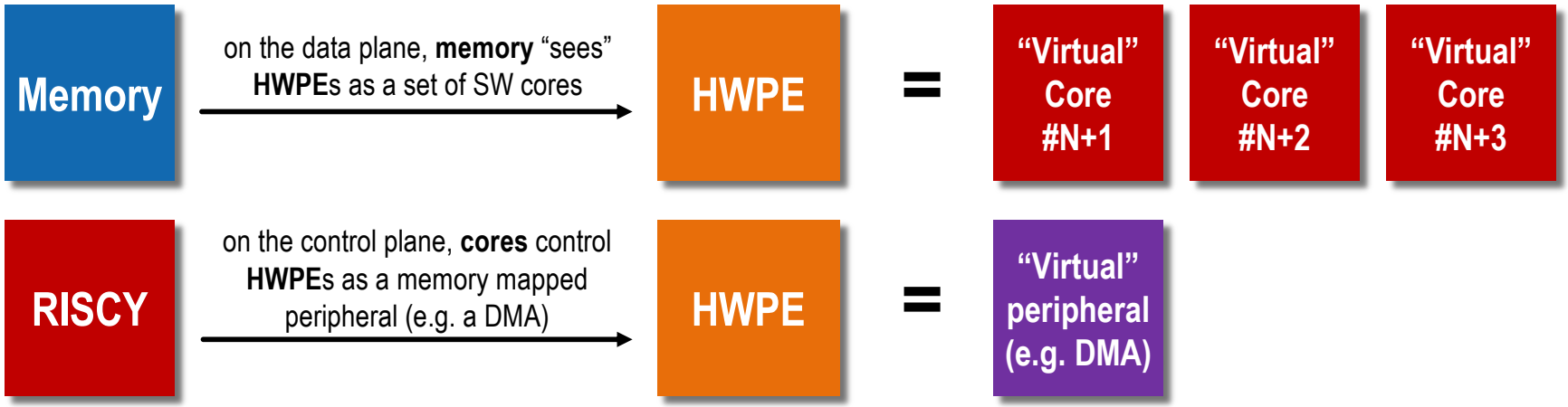
Hardware Accelerators from a PULP-y Perspective (3)

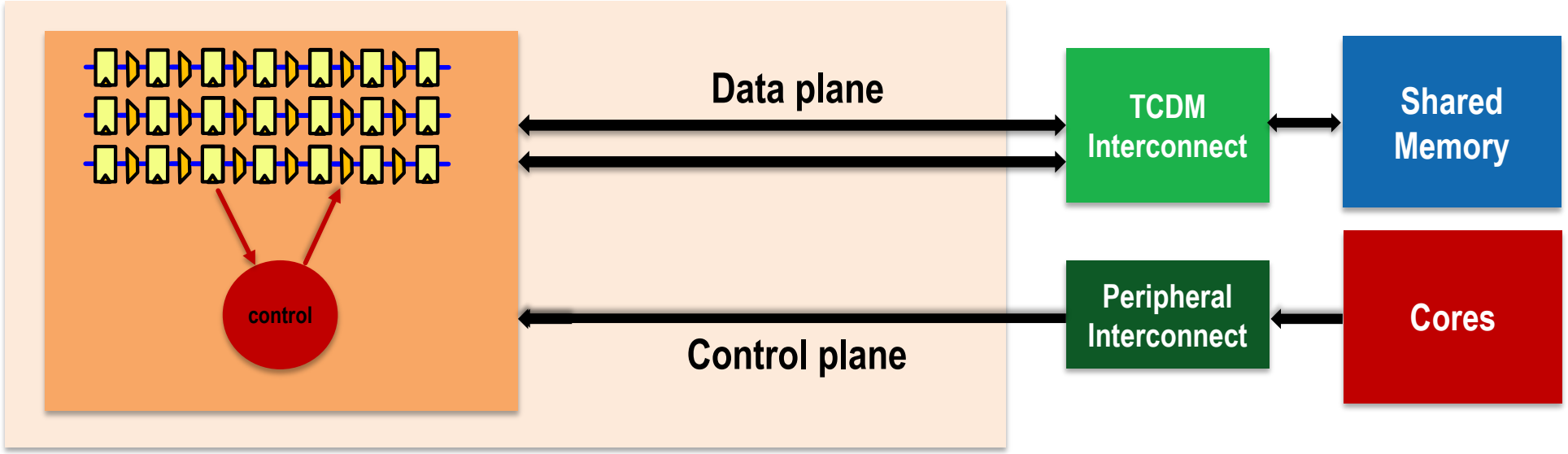


Hardware Accelerators from a PULP-y Perspective (3)

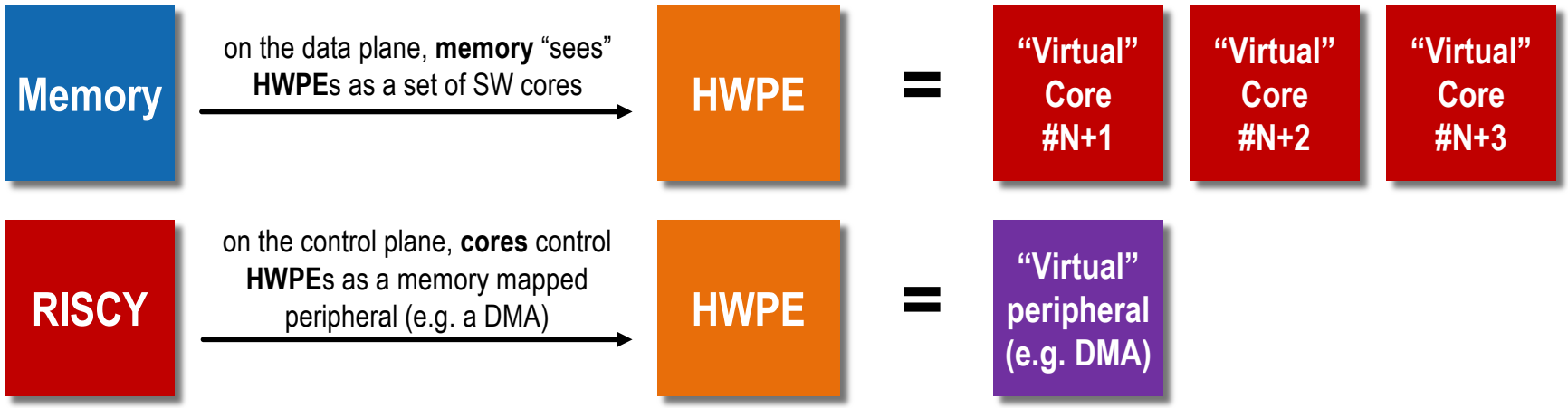


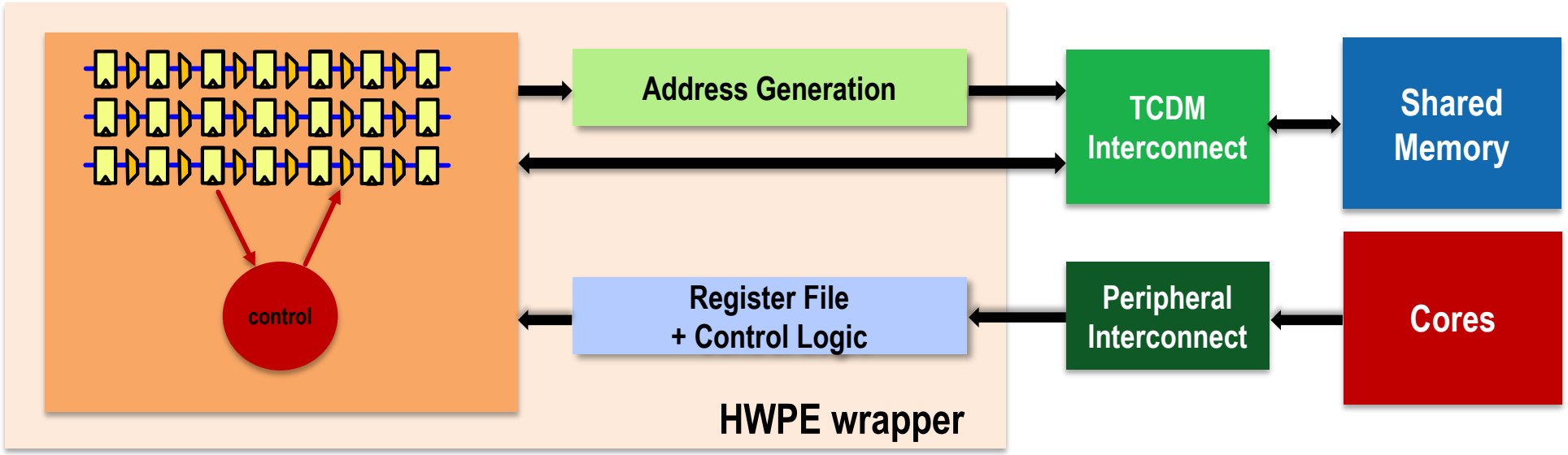
Hardware Processing Engines (HWPEs)



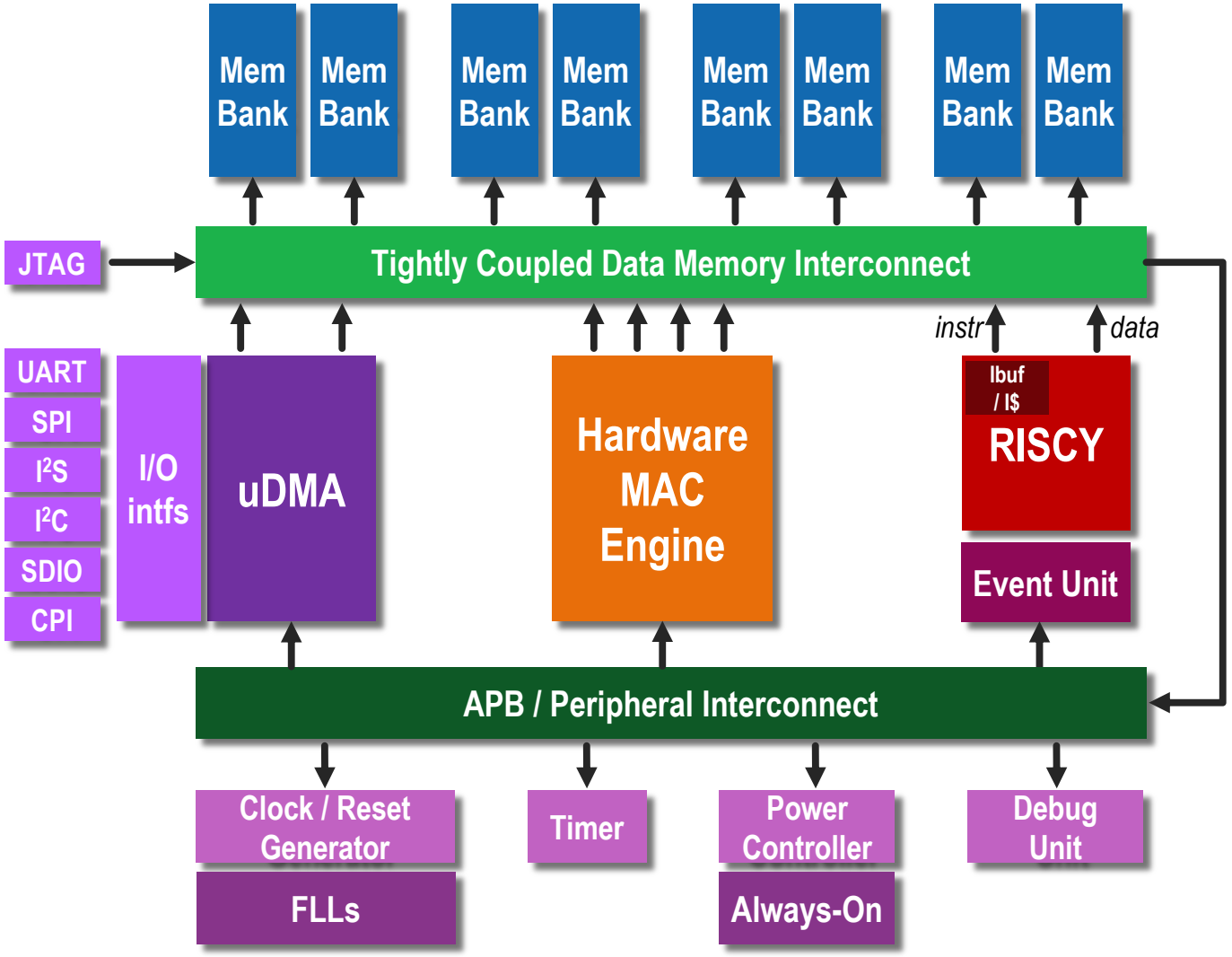


Hardware Processing Engines (HWPEs)

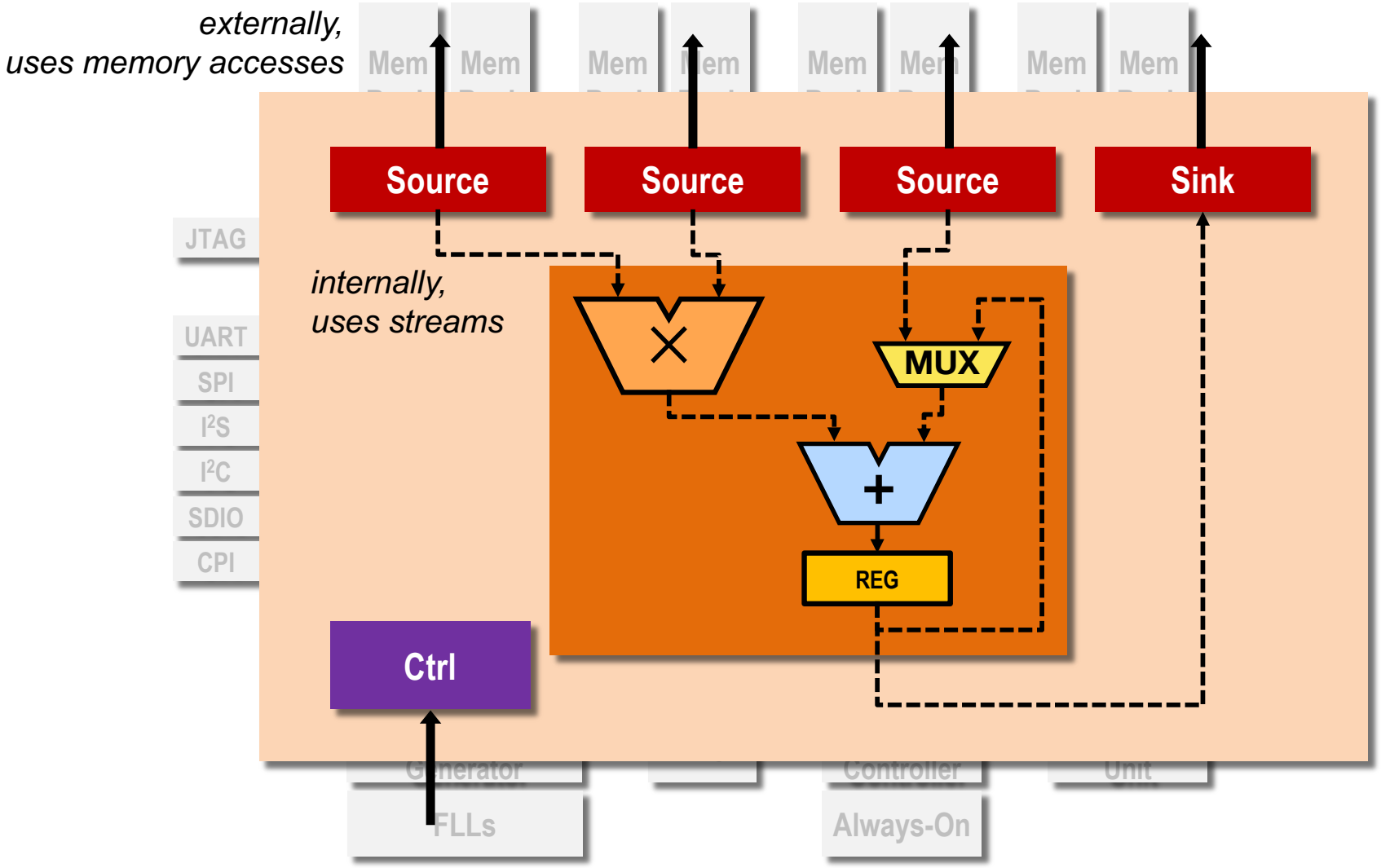




Mini-Demo (from PULPissimo)



Mini-Demo (from PULPissimo)



So far so good…

- … but the **MAC Engine** is not much more than a toy!
  - very **simple behavior**
  - < **700 lines** of SystemVerilog code for RTL
  - **~100 lines** of C code for software test
  - deliberately suboptimal design:
    - more source / sink units than needed
    - support for a single hardware loop
  - designed as an **example of HWPE** for the open-source release

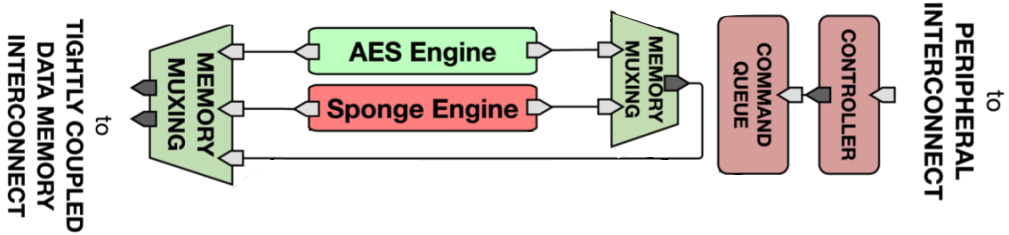
- In the next part of the talk, we focus on ***real*** processing engines:
  - **Crypto Engine** for AES / Keccak-f400 encryption
  - **Convolution Engine** for Convolutional Neural Network *inference*
  - **NST / NTX** for Deep Neural Network inference and *learning*

Internet of Things

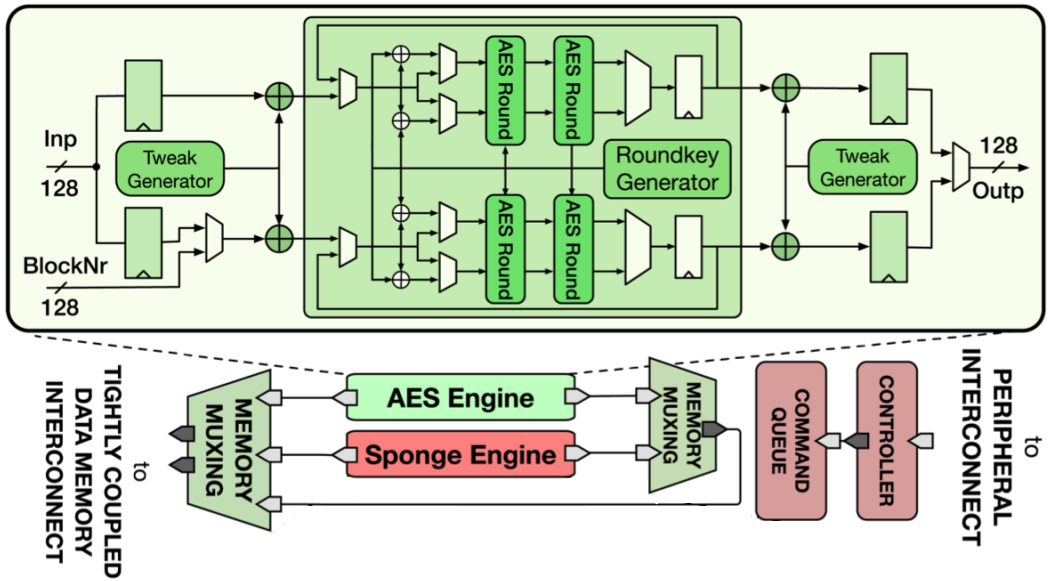
High-Performance Computing

HWCrypt – a Crypto Primitive Accelerator

HWCrypt is a «collection» of two crypto engines plugged to the *shared memory* and controlled via the *periph interconnect*



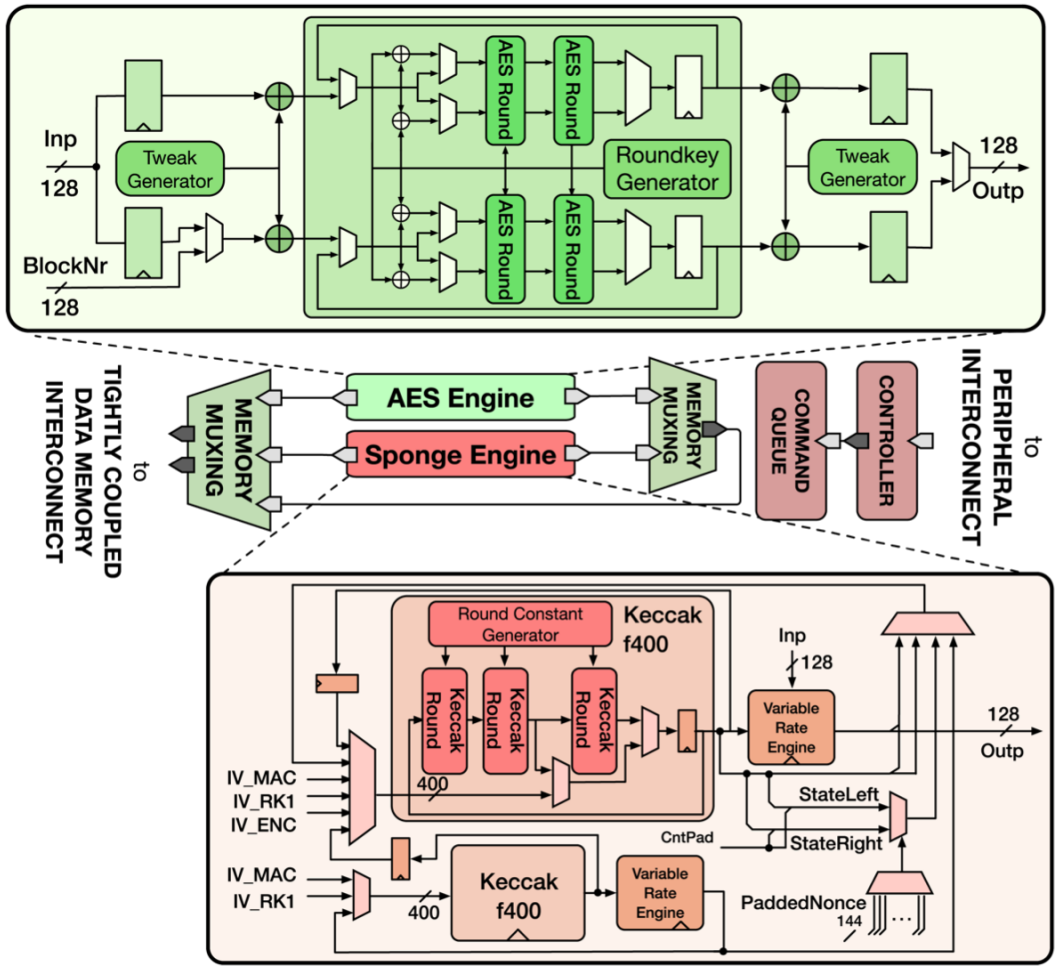
HWCrypt – a Cryptographic Accelerator



HWCrypt is a «collection» of two crypto engines plugged to the *shared memory* and controlled via the *periph interconnect*

- **AES Engine**
  - *AES-128-ECB*: fast but not secure (plaintext patterns are ~visible in ciphertext)– for comparison!
  - *AES-128-XTS*: each block encrypted with a different tweak – just as fast in the HWCrypt
  - individual execution of cipher rounds (to speed up new SW-based AES-based algorithms)

HWCrypt – a Cryptographic Accelerator



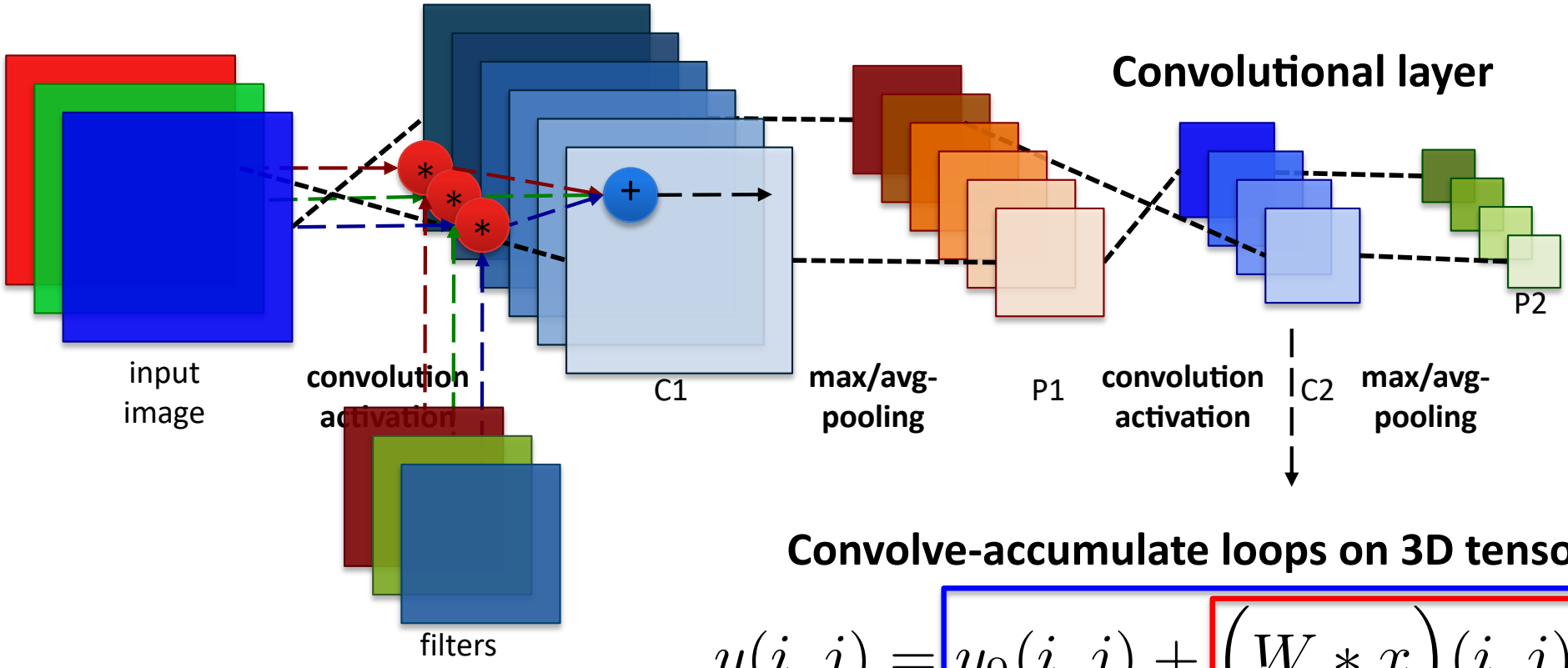
HWCrypt is a «collection» of two crypto engines plugged to the *shared memory* and controlled via the *periph interconnect*

- **AES Engine**
  - *AES-128-ECB*: fast but not secure (plaintext patterns are ~visible in ciphertext)– for comparison!
  - *AES-128-XTS*: each block encrypted with a different tweak – just as fast in the HWCrypt
  - individual execution of cipher rounds (to speed up new SW-based AES-based algorithms)
- **Sponge Engine**
  - two instances of *Keccak-f[400]*
  - leakage-resilient encryption scheme [1]
  - similar performance to AES engine

[1] T. Unterluggauer et al., Leakage Bounds for Gaussian Side Channels, CARDIS 2017

HWCE – a Deep Neural Inference Accelerator

HW Convolution Engine [2] performs sliding-window based convolution/accumulation targeted at **deep neural inference** on low-power devices, using **16bit fixed-point data**



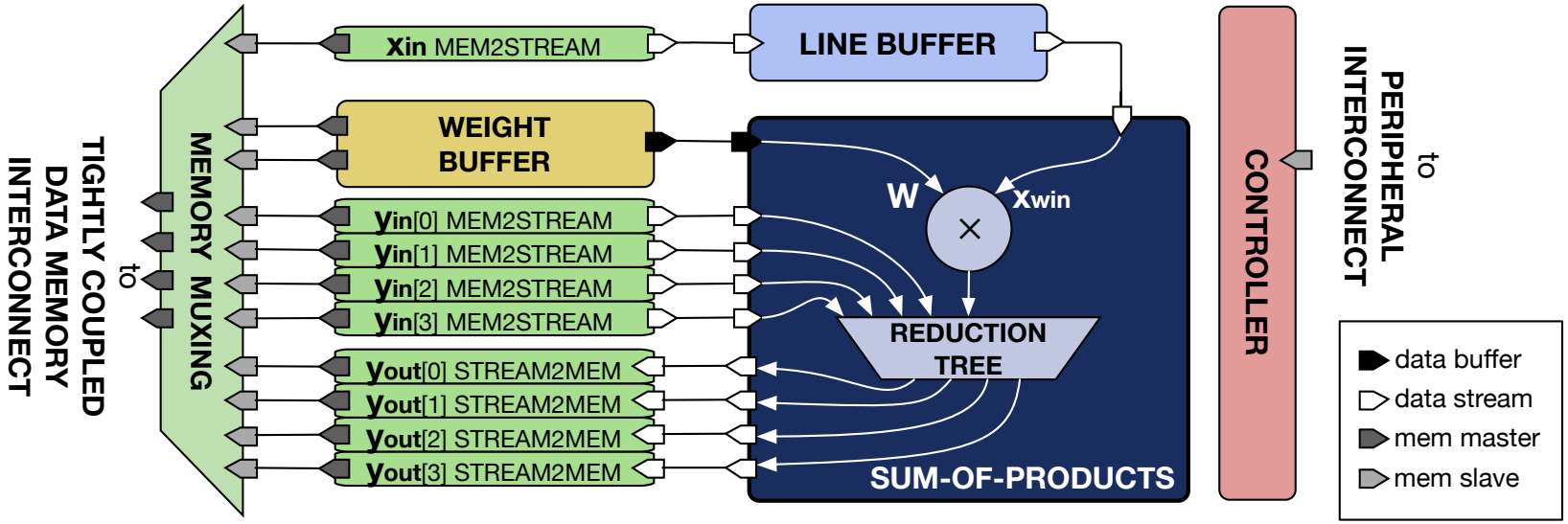
Convolve-accumulate loops on 3D tensors

$$y(i,j) = y_0(i,j) + \Big(W * x\Big)(i,j)$$

accumulate convolve

HWCE – a Deep Neural Inference Accelerator

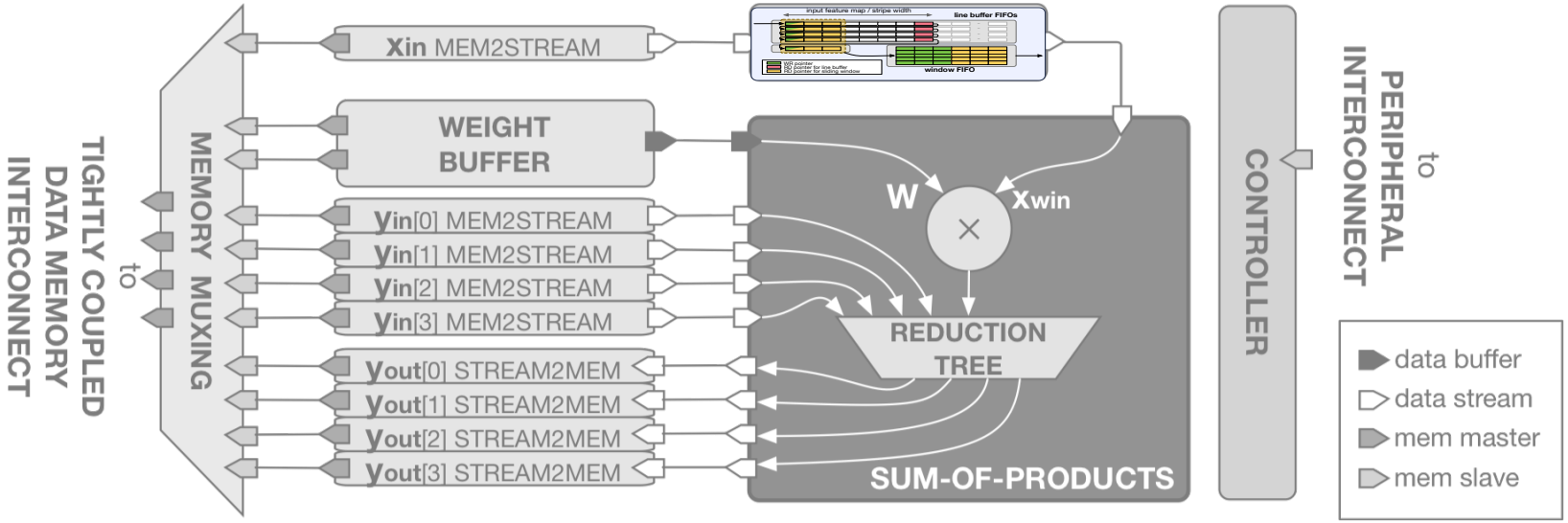
HW Convolution Engine [2] performs sliding-window based convolution/accumulation targeted at **deep neural inference** on low-power devices, using **16bit fixed-point data**



[2] F. Conti and L. Benini, A Ultra-Low-Energy Convolution Engine for Fast Brain-Inspired Vision in Multicore Clusters, DATE 2015

HWCE – a Deep Neural Inference Accelerator

HW Convolution Engine [2] performs sliding-window based convolution/accumulation targeted at **deep neural inference** on low-power devices, using **16bit fixed-point data**

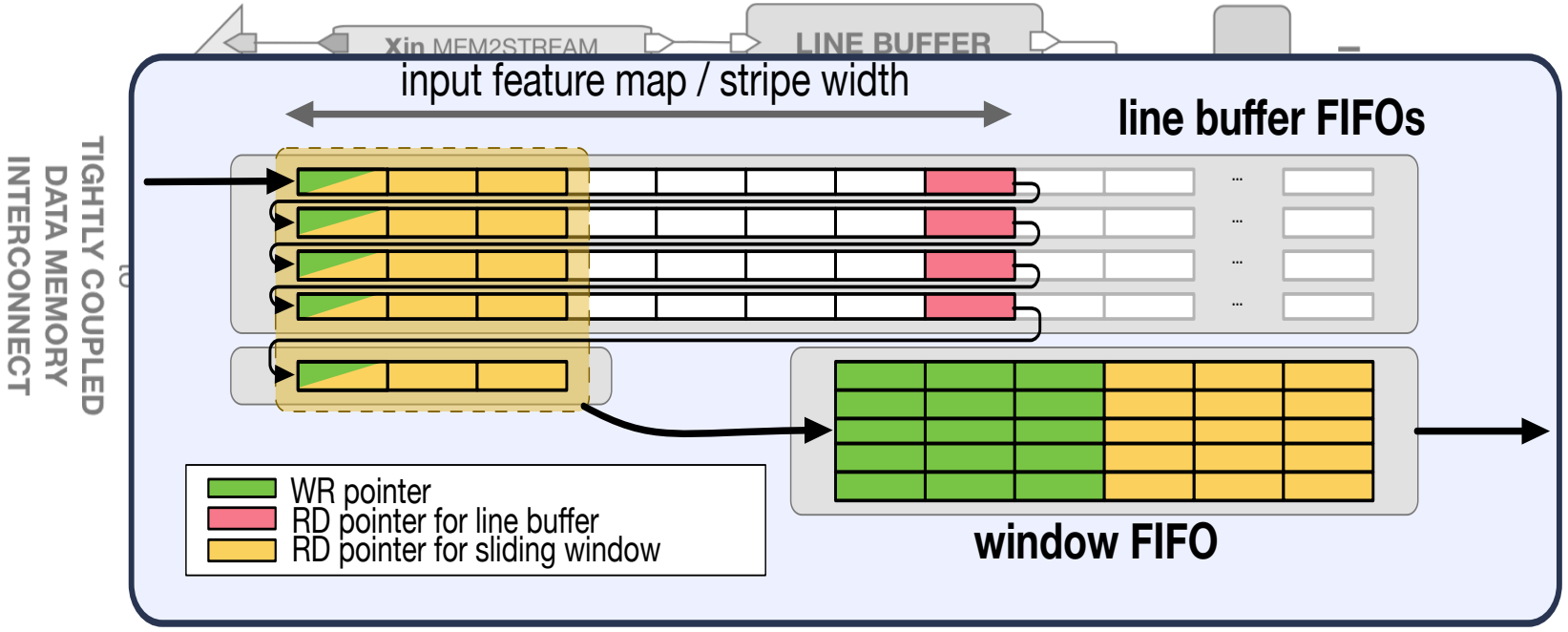


[2] F. Conti and L. Benini, A Ultra-Low-Energy Convolution Engine for Fast Brain-Inspired Vision in Multicore Clusters, DATE 2015

HWCE – a Deep Neural Inference Accelerator

HW Convolution Engine [2] performs sliding-window based convolution/accumulation targeted at **deep neural inference** on low-power devices, using **16bit fixed-point data**

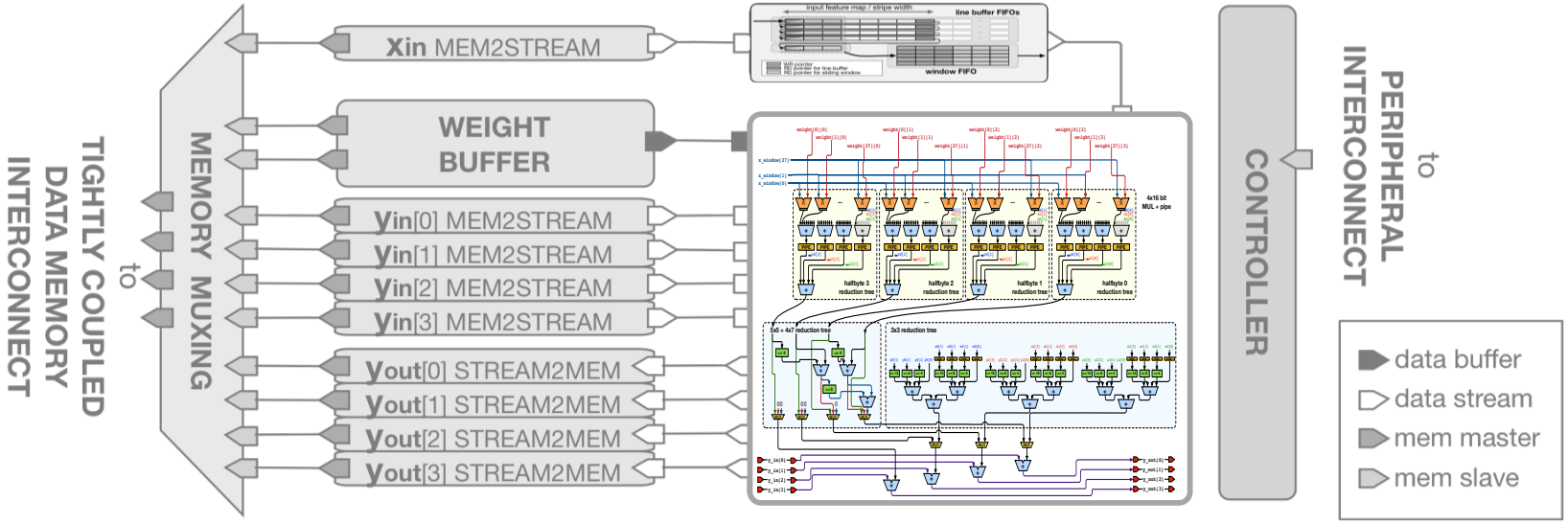
*Configurable **line buffer** to extract **one 5x5 / 4x7** or up to **three 3x3** sliding windows*



[2] F. Conti and L. Benini, A Ultra-Low-Energy Convolution Engine for Fast Brain-Inspired Vision in Multicore Clusters, DATE 2015

HWCE – a Deep Neural Inference Accelerator

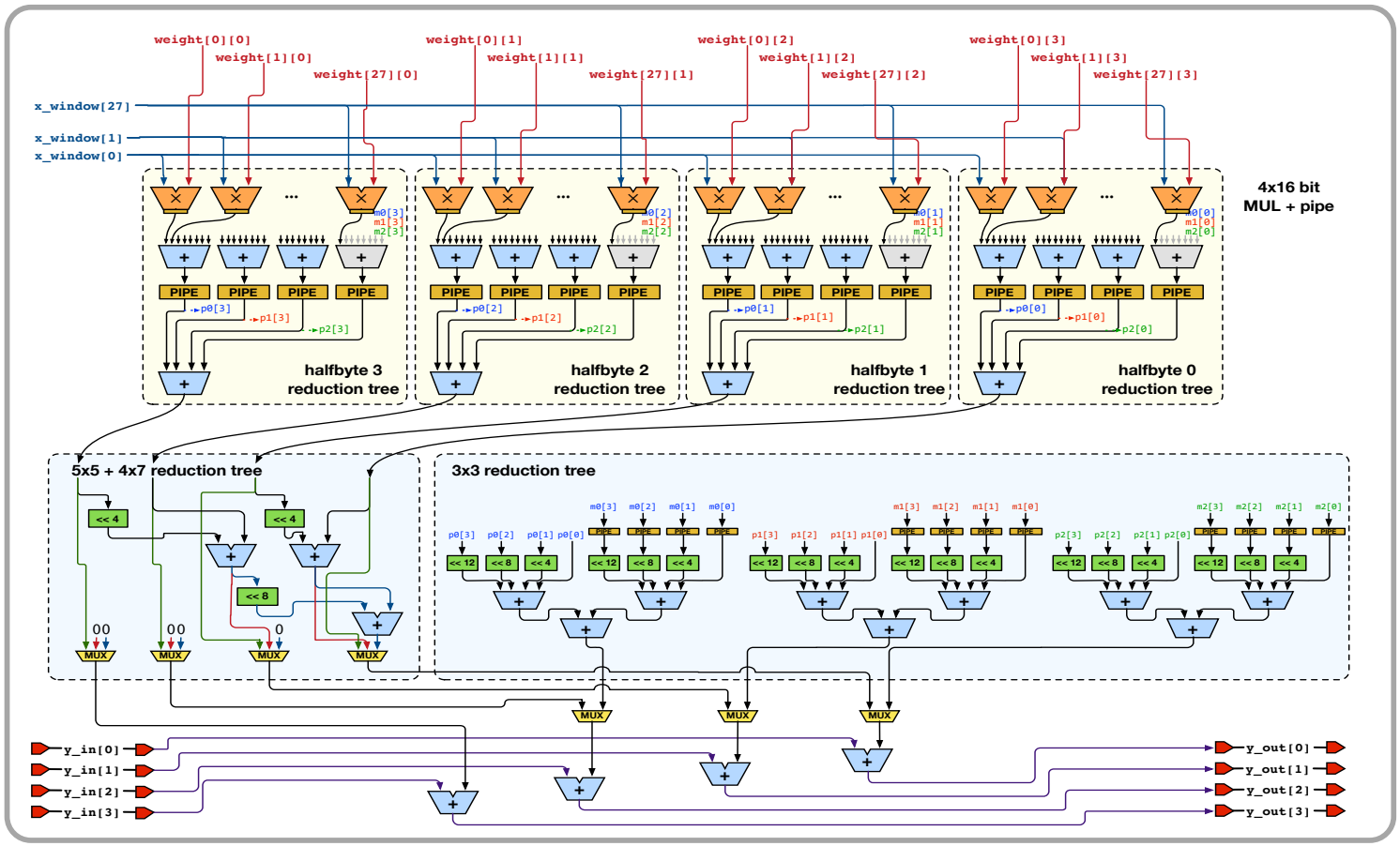
HW Convolution Engine [2] performs sliding-window based convolution/accumulation targeted at **deep neural inference** on low-power devices, using **16bit fixed-point data**



[2] F. Conti and L. Benini, A Ultra-Low-Energy Convolution Engine for Fast Brain-Inspired Vision in Multicore Clusters, DATE 2015

HWCE – a Deep Neural Inference Accelerator

HW Convolution Engine [2] performs sliding-window based convolution/accumulation targeted at **deep neural inference** on low-power devices, using **16bit fixed-point data**

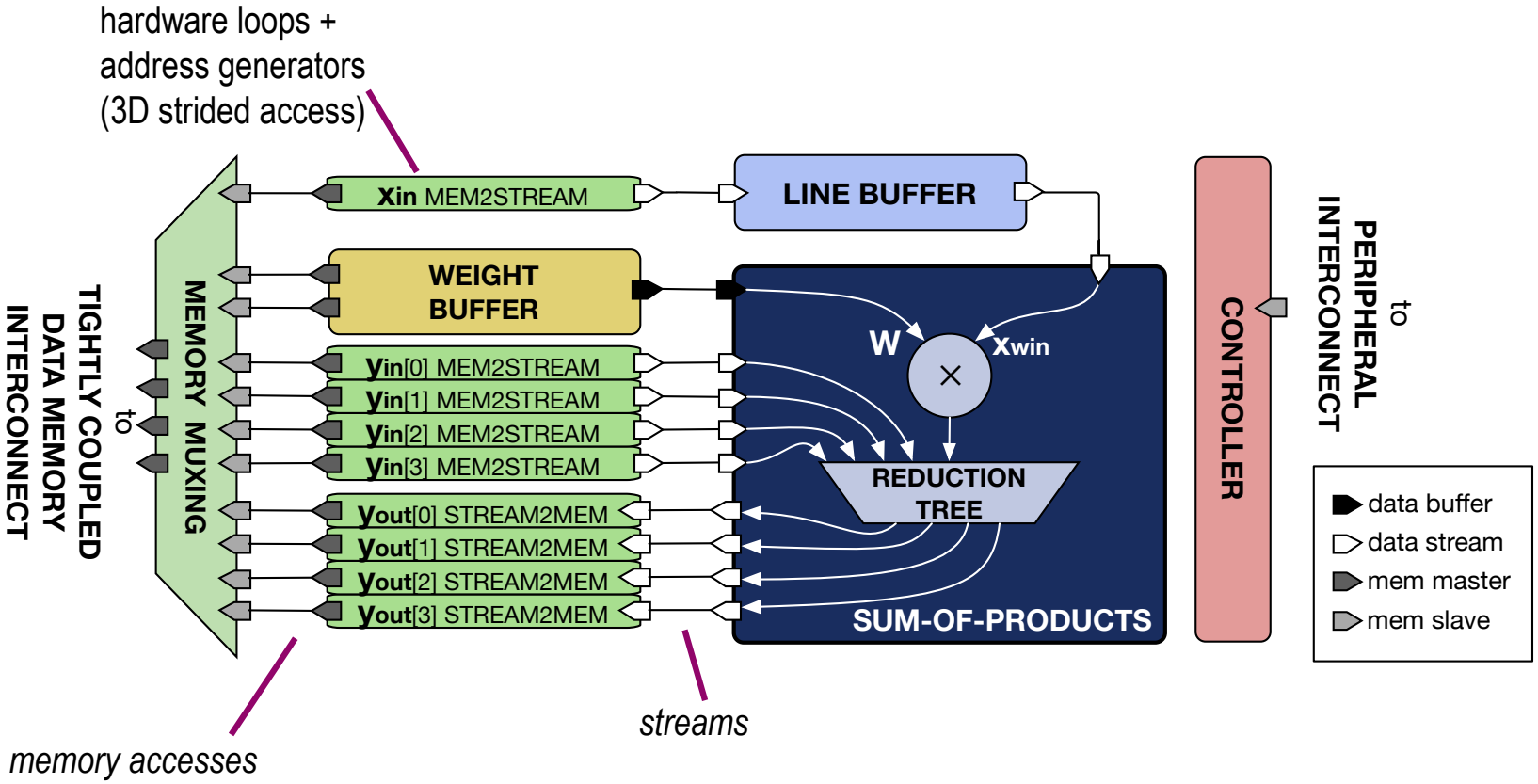


*Datapath with **28 mults** optimized to provide full throughput on three 3x3 windows, or one 5x5 – 4x7 window*

[2] F. Conti and L. Benini, A Ultra-Low-Energy Convolution Engine for Fast Brain-Inspired Vision in Multicore Clusters, DATE 2015

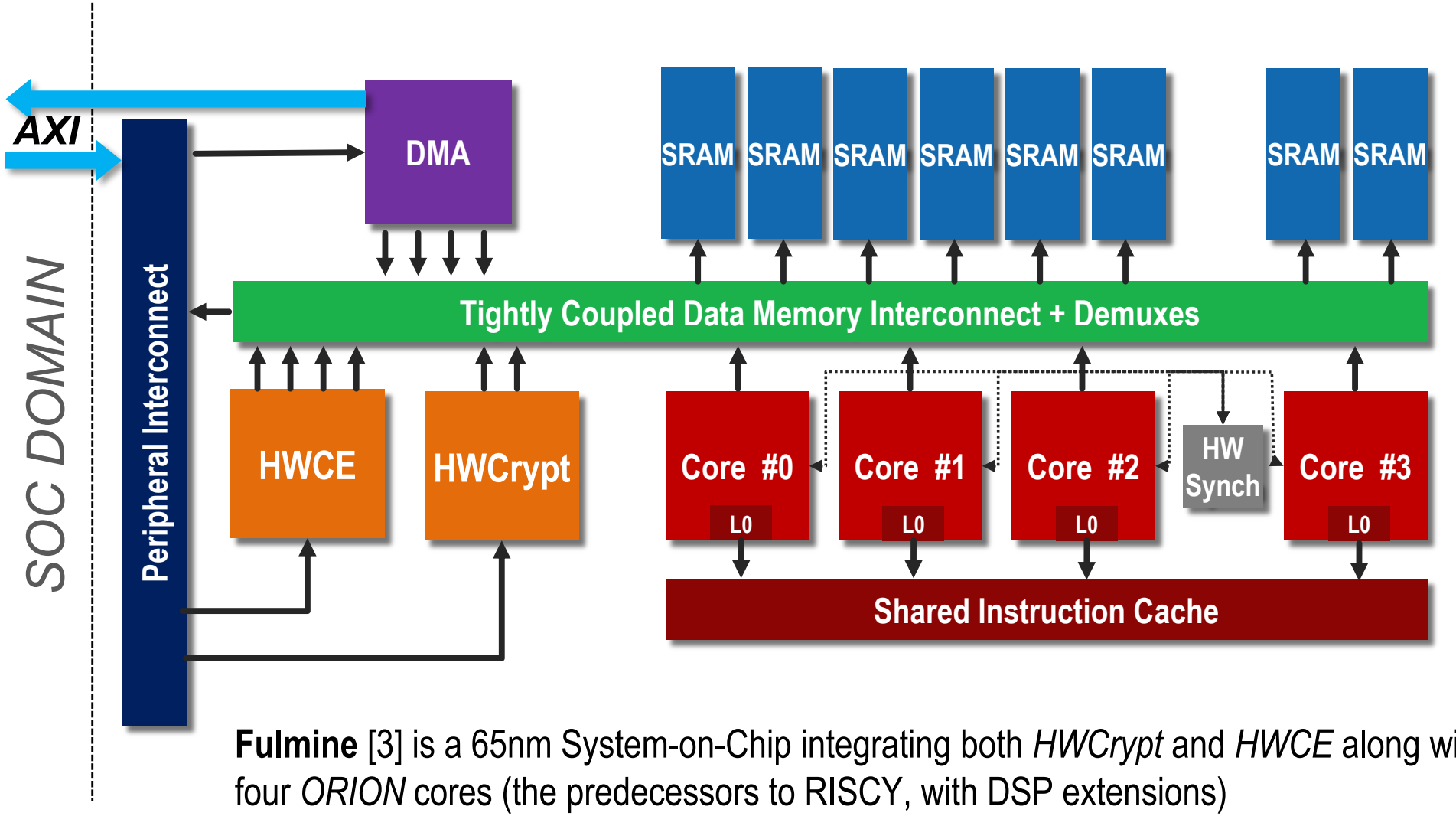
HWCE – a Deep Neural Inference Accelerator

HW Convolution Engine [2] performs sliding-window based convolution/accumulation targeted at **deep neural inference** on low-power devices, using **16bit fixed-point data**



[2] F. Conti and L. Benini, A Ultra-Low-Energy Convolution Engine for Fast Brain-Inspired Vision in Multicore Clusters, DATE 2015

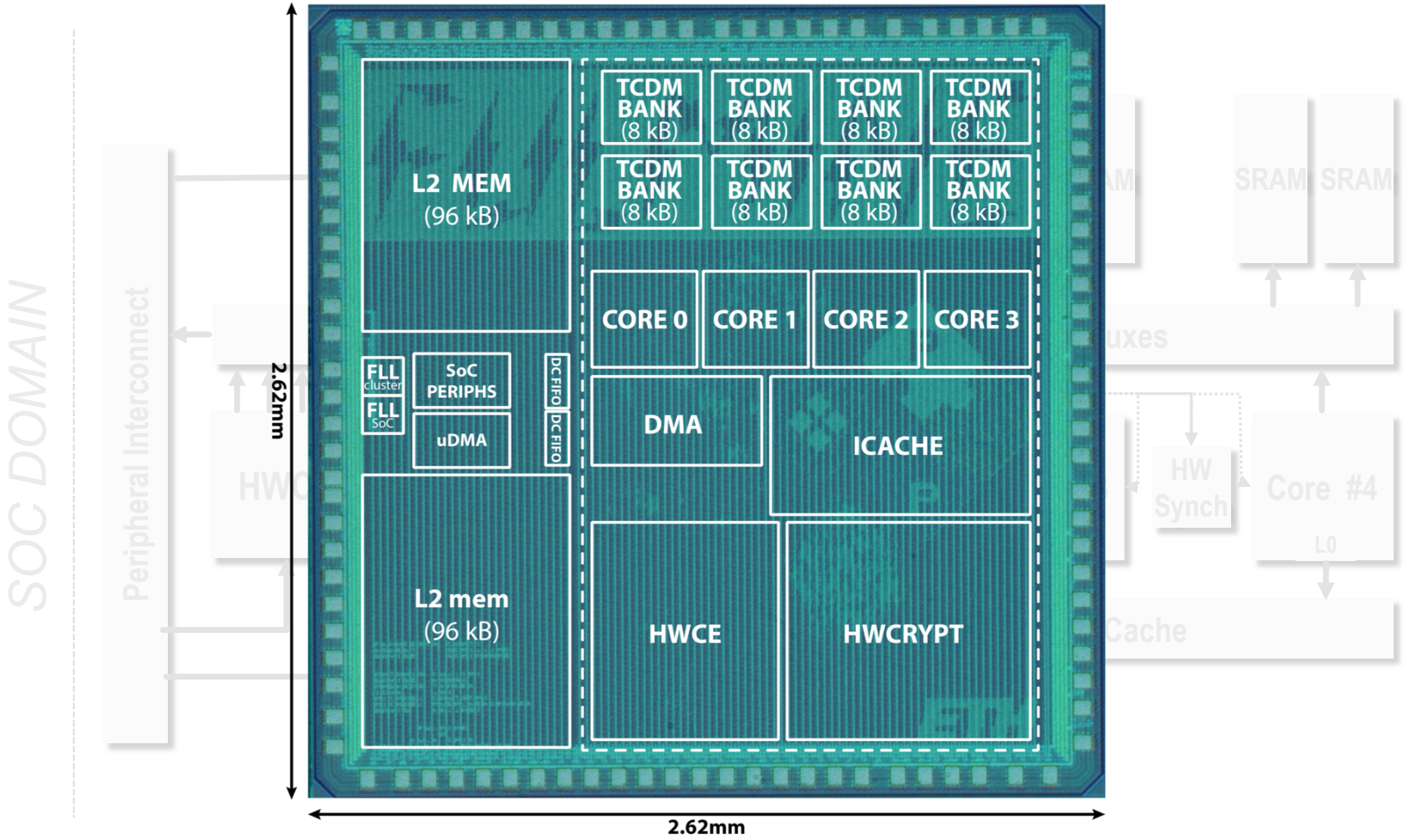
Fulmine: a HW-Accelerated IoT System-on-Chip



Fulmine [3] is a 65nm System-on-Chip integrating both *HWCrypt* and *HWCE* along with four *ORION* cores (the predecessors to RISCY, with DSP extensions)

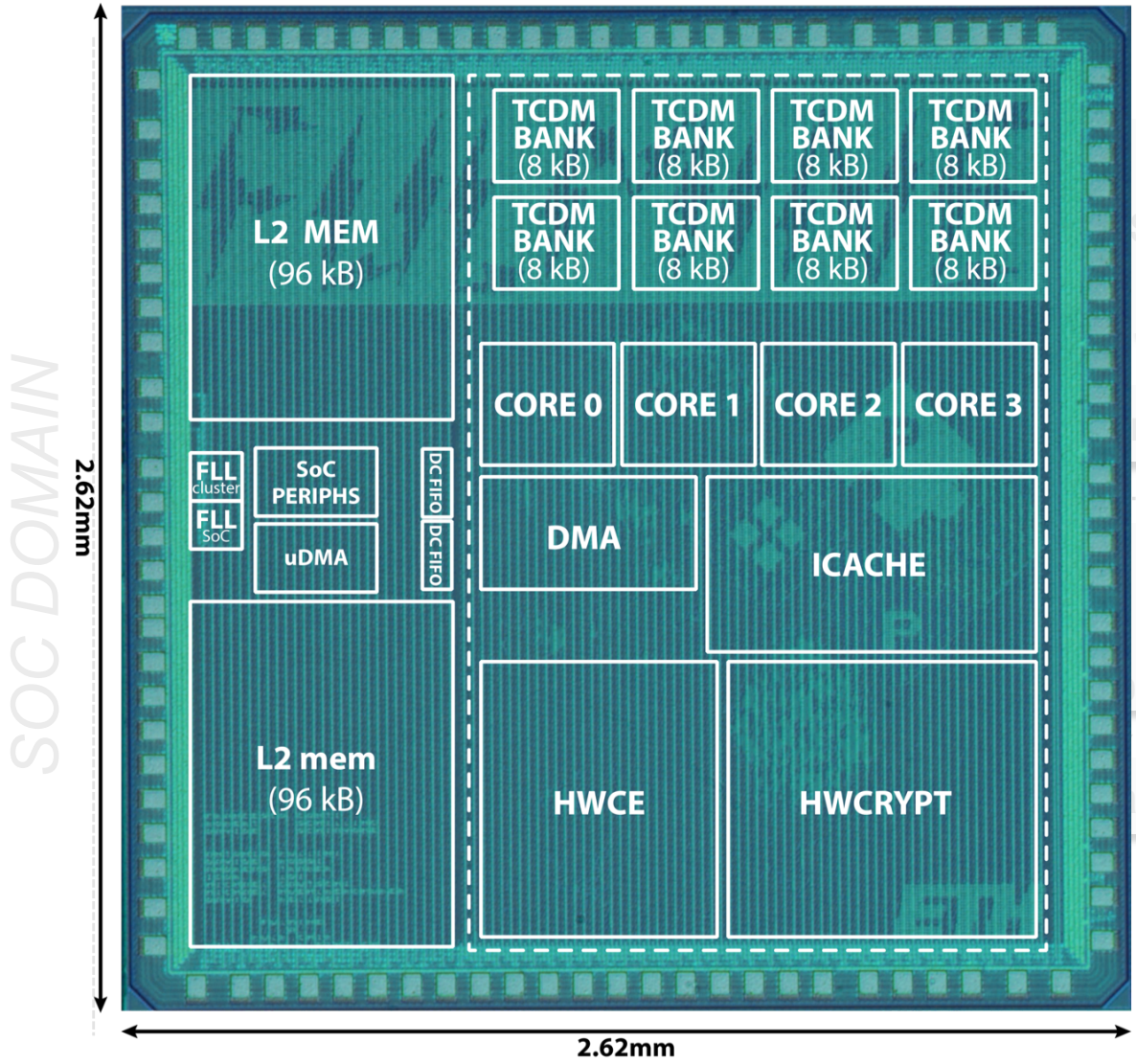
[3] F. Conti et al., An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics, IEEE TCAS-I 2017

Fulmine: a HW-Accelerated IoT System-on-Chip



[3] F. Conti et al., An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics, IEEE TCAS-I 2017

Fulmine: a HW-Accelerated IoT System-on-Chip



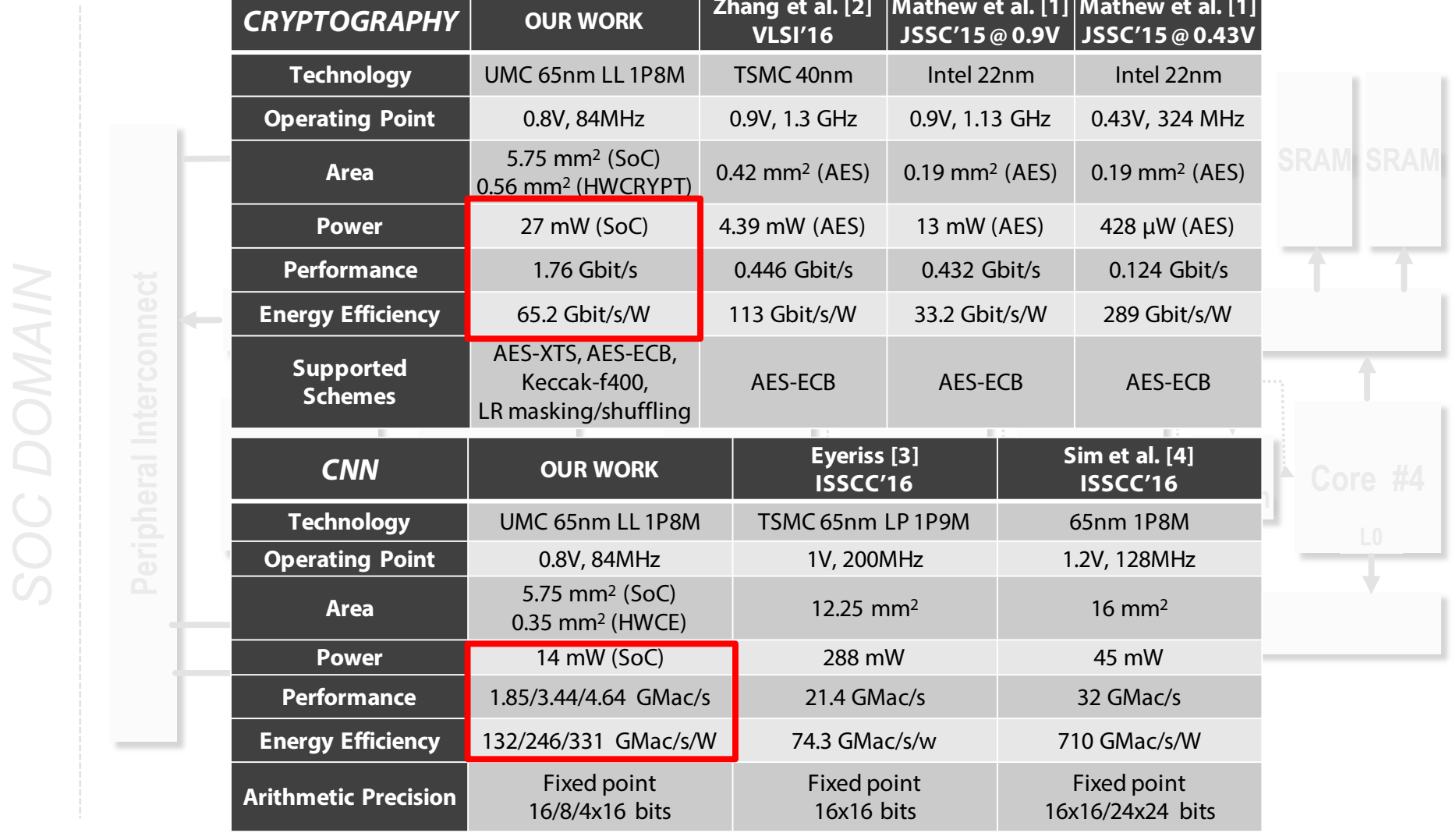
- UMC 65nm technology
  - 6.86 mm^2
- 4 cores, 2 accelerators
  - **HWCE** for 3D conv layers
  - **HWCRYPT** for AES
  - **DSP-optimized** cores
- 64 kB of L1, 192 kB of L2
- First version of *uDMA* for I/O with no SW intervention
  - QSPI master/slave
  - I^{2}C
  - I^{2}S
  - UART
- Working on board (**demo** if there's time…)

[3] F. Conti et al., An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics, IEEE TCAS-I 2017

Fulmine: a HW-Accelerated IoT System-on-Chip

CRYPTOGRAPHY	OUR WORK	Zhang et al. [2] VLSI'16	Mathew et al. [1] JSSC'15 @ 0.9V	Mathew et al. [1] JSSC'15 @ 0.43V
Technology	UMC 65nm LL 1P8M	TSMC 40nm	Intel 22nm	Intel 22nm
Operating Point	0.8V, 84MHz	0.9V, 1.3 GHz	0.9V, 1.13 GHz	0.43V, 324 MHz
Area	5.75 mm^2 (SoC) 0.56 mm^2 (HWCRYPT)	0.42 mm^2 (AES)	0.19 mm^2 (AES)	0.19 mm^2 (AES)
Power	27 mW (SoC)	4.39 mW (AES)	13 mW (AES)	428 µW (AES)
Performance	1.76 Gbit/s	0.446 Gbit/s	0.432 Gbit/s	0.124 Gbit/s
Energy Efficiency	65.2 Gbit/s/W	113 Gbit/s/W	33.2 Gbit/s/W	289 Gbit/s/W
Supported Schemes	AES-XTS, AES-ECB, Keccak-f400, LR masking/shuffling	AES-ECB	AES-ECB	AES-ECB

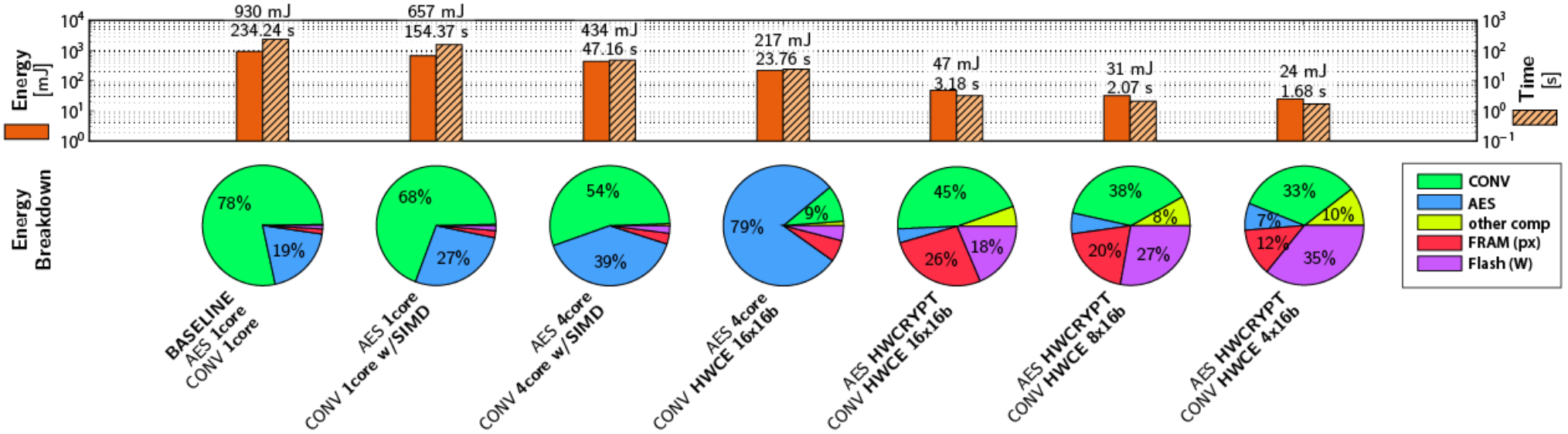
CNN	OUR WORK	Eyeriss [3] ISSCC'16	Sim et al. [4] ISSCC'16
Technology	UMC 65nm LL 1P8M	TSMC 65nm LP 1P9M	65nm 1P8M
Operating Point	0.8V, 84MHz	1V, 200MHz	1.2V, 128MHz
Area	5.75 mm^2 (SoC) 0.35 mm^2 (HWCE)	12.25 mm^2	16 mm^2
Power	14 mW (SoC)	288 mW	45 mW
Performance	1.85/3.44/4.64 GMac/s	21.4 GMac/s	32 GMac/s
Energy Efficiency	132/246/331 GMac/s/W	74.3 GMac/s/w	710 GMac/s/W
Arithmetic Precision	Fixed point 16/8/4x16 bits	Fixed point 16x16 bits	Fixed point 16x16/24x24 bits



[3] F. Conti et al., An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics, IEEE TCAS-I 2017

Fulmine: a HW-Accelerated IoT System-on-Chip

ResNet-18 + AES encryption of partial results
Fulmine silicon measurements [3] for CONV, AES, DMA + datasheet values for COTS FRAM, Flash



[3] F. Conti et al., An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics, IEEE TCAS-I 2017

GAP8: a commercial HW-Accelerated PULP

- HW Acceleration ecosystem for PULP is starting to be **mature**
- **Greenwaves Technologies** developed the first commercial product based on a HW-accelerated PULP system
  - **9** DSP-enhanced and customized **RISCY** cores (open source)
  - **1** customized **HW Convolution Engine** (under non-exclusive license)
- HWCE can compute more than **2.2 Gop/s** in **<10mW** at nominal voltage
  - energy reduction with respect to 8 SW cores of **~14x** within a lower power envelope (<**10mW** vs **25mW**)

PULP in HPC?

- So far, focus on **low-power** applications: nano-UAVs, near-sensor processing, etc…
- ... what about **high-performance computing**?
  - energy efficiency of capital importance (**power = $$$** spent for **cooling**, **energy bill**)
  - next talk will focus on HPC-oriented architectures

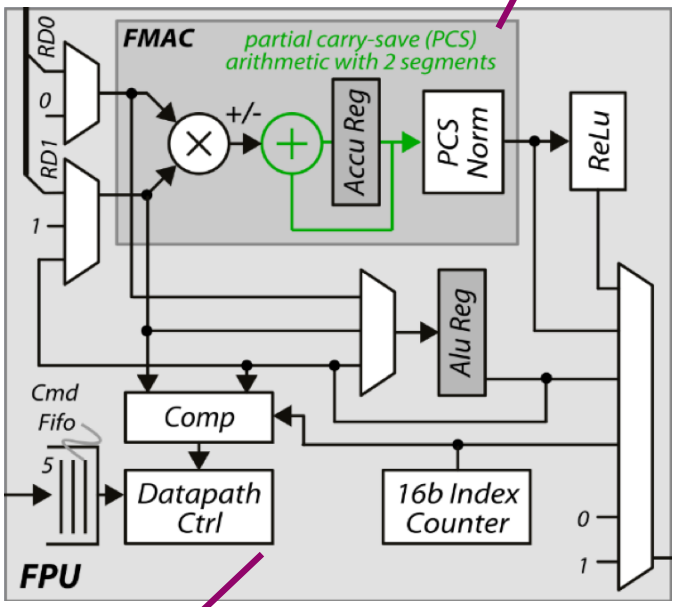
- **Hardware accelerators** provide a key technology for HPC
  - compute-dominated workloads
  - highly parallel workloads
  - efficiency in Joules/op and power envelope in kW are important metrics

- 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.



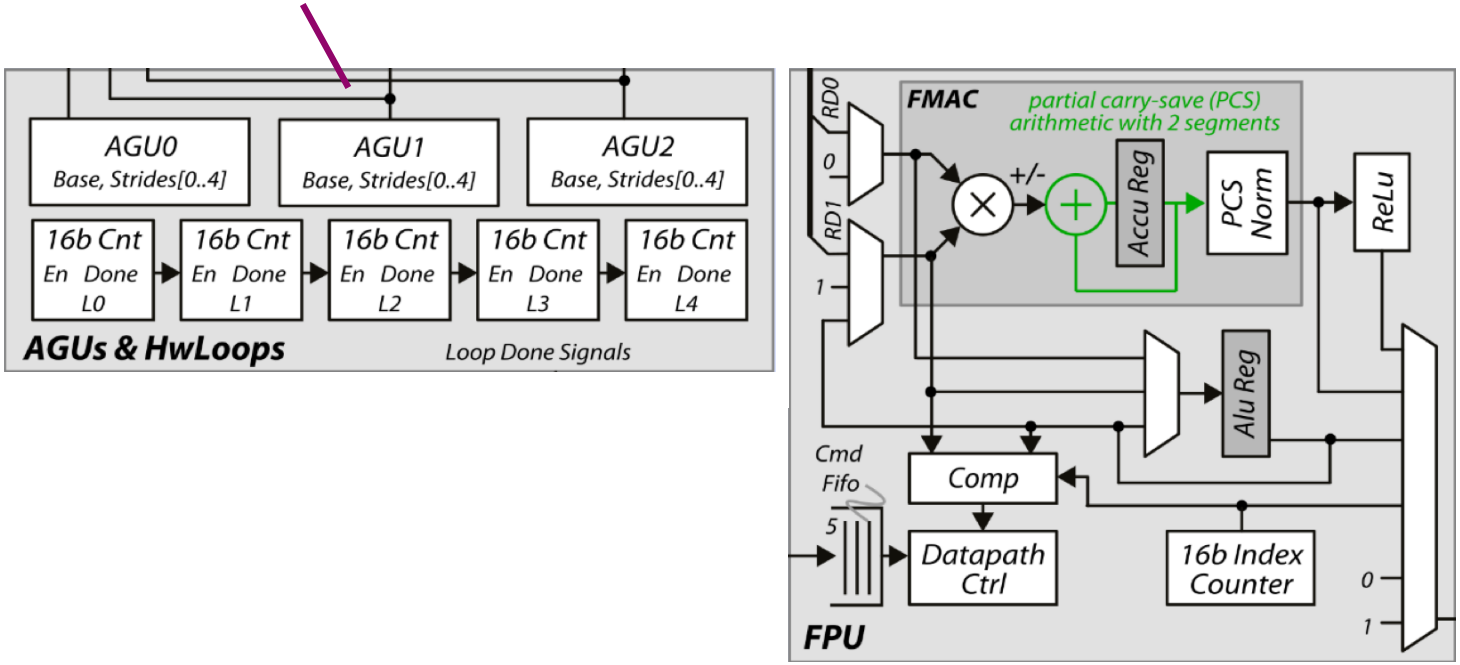
Datapath supports additional ReLU, comparison, and masking operations (for DNN layer derivatives)

[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

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…)

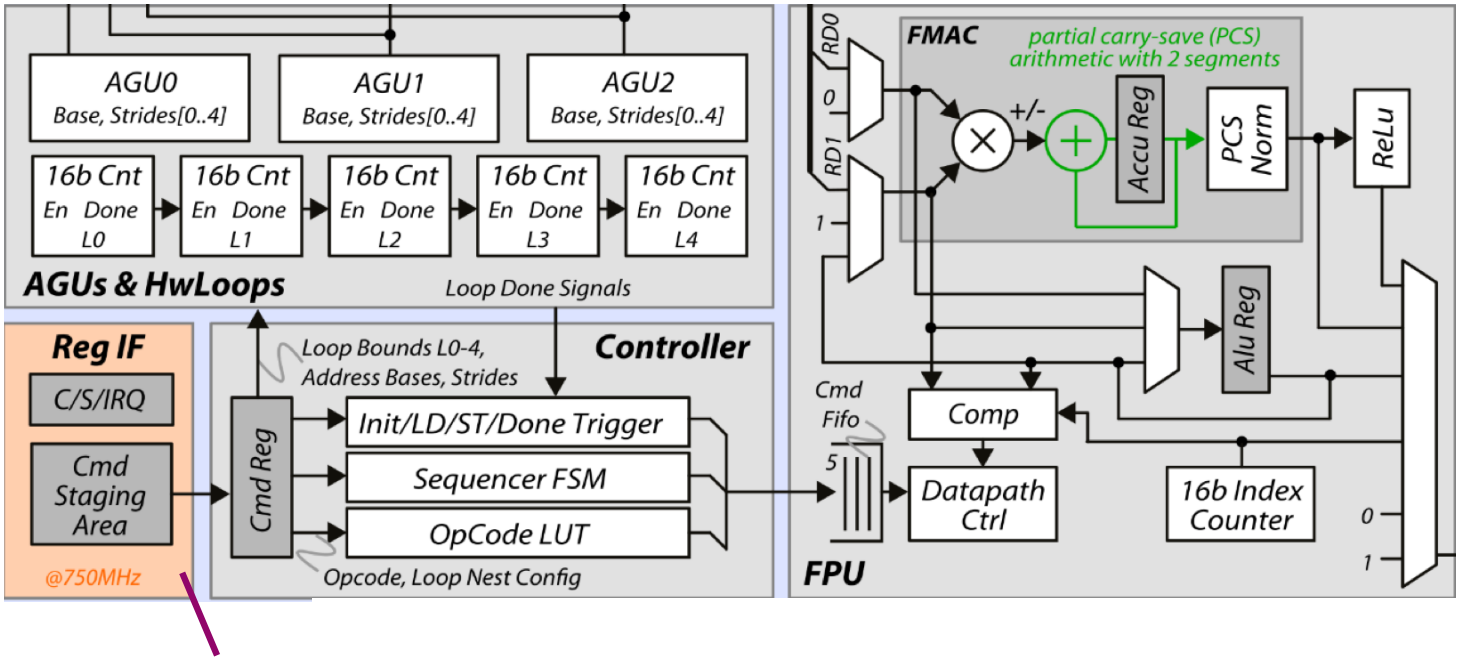
5 nested hardware loops and three address generators



[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

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…)

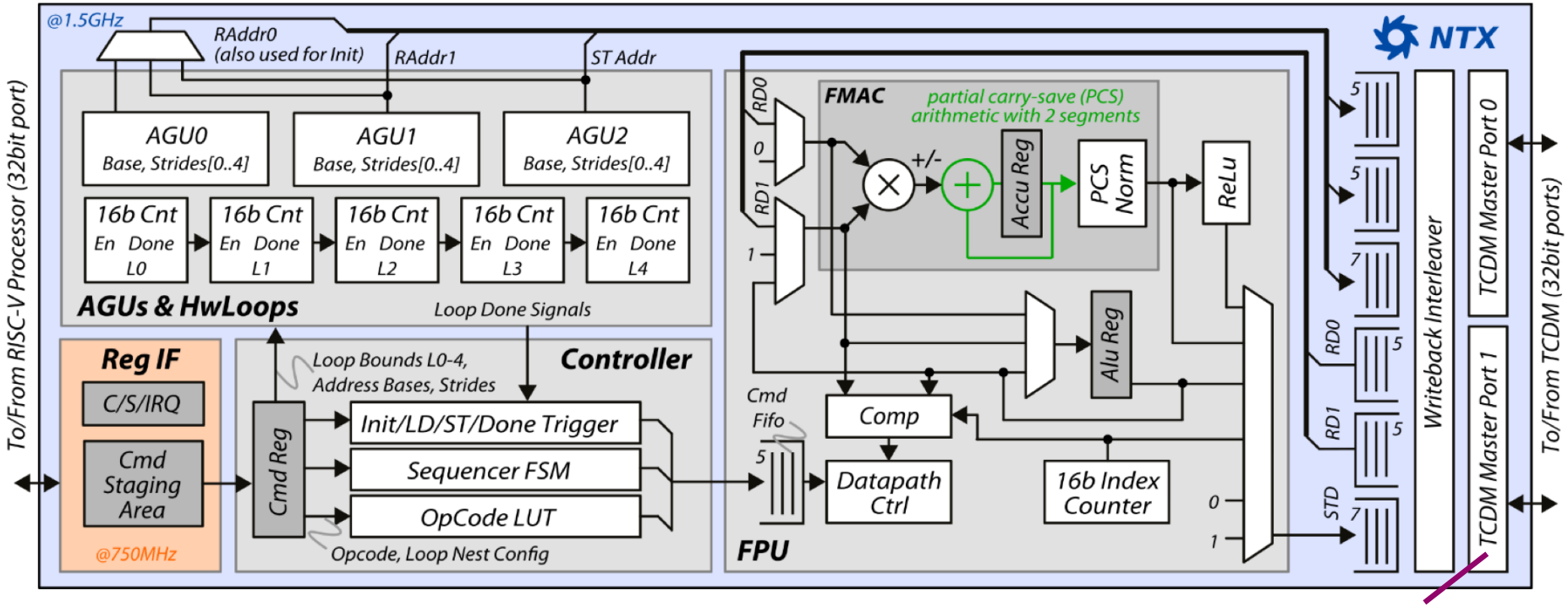


Memory-mapped control

[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

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…)

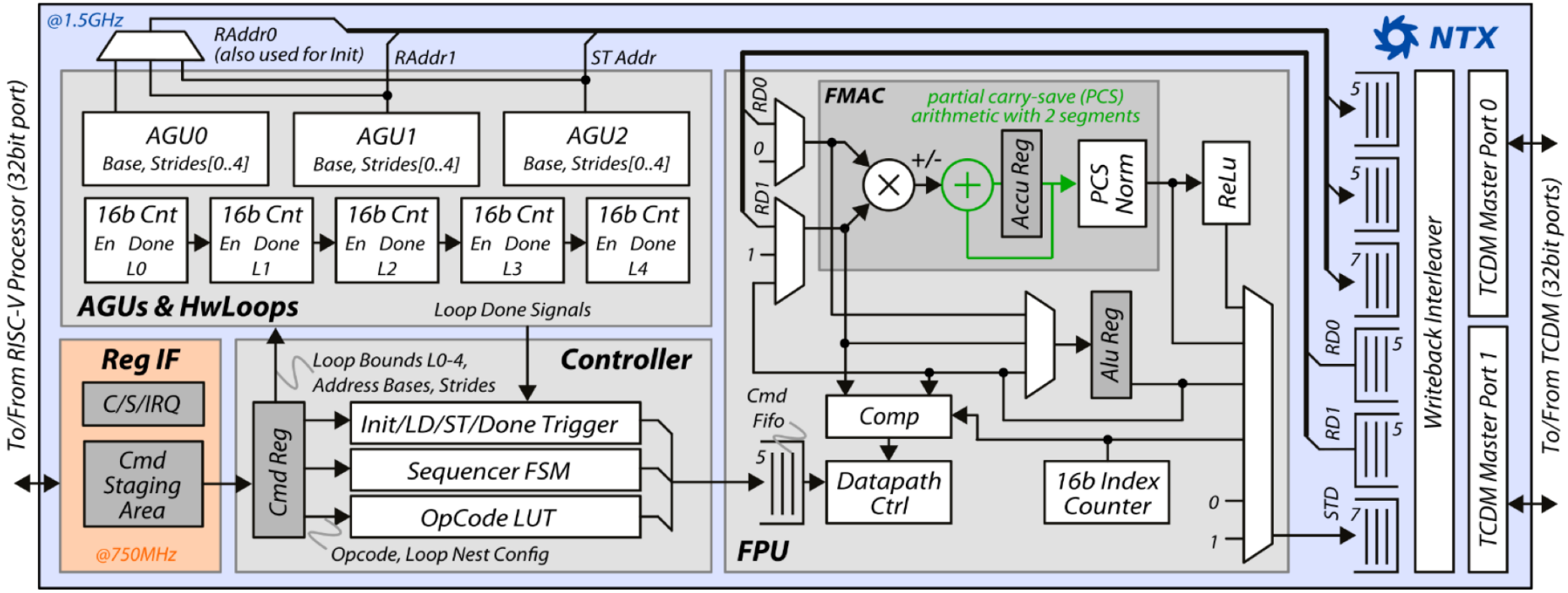


2 ports into cluster memory read operands and write back results.

[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

NTX: Boosting HWPEs for Deep Learning

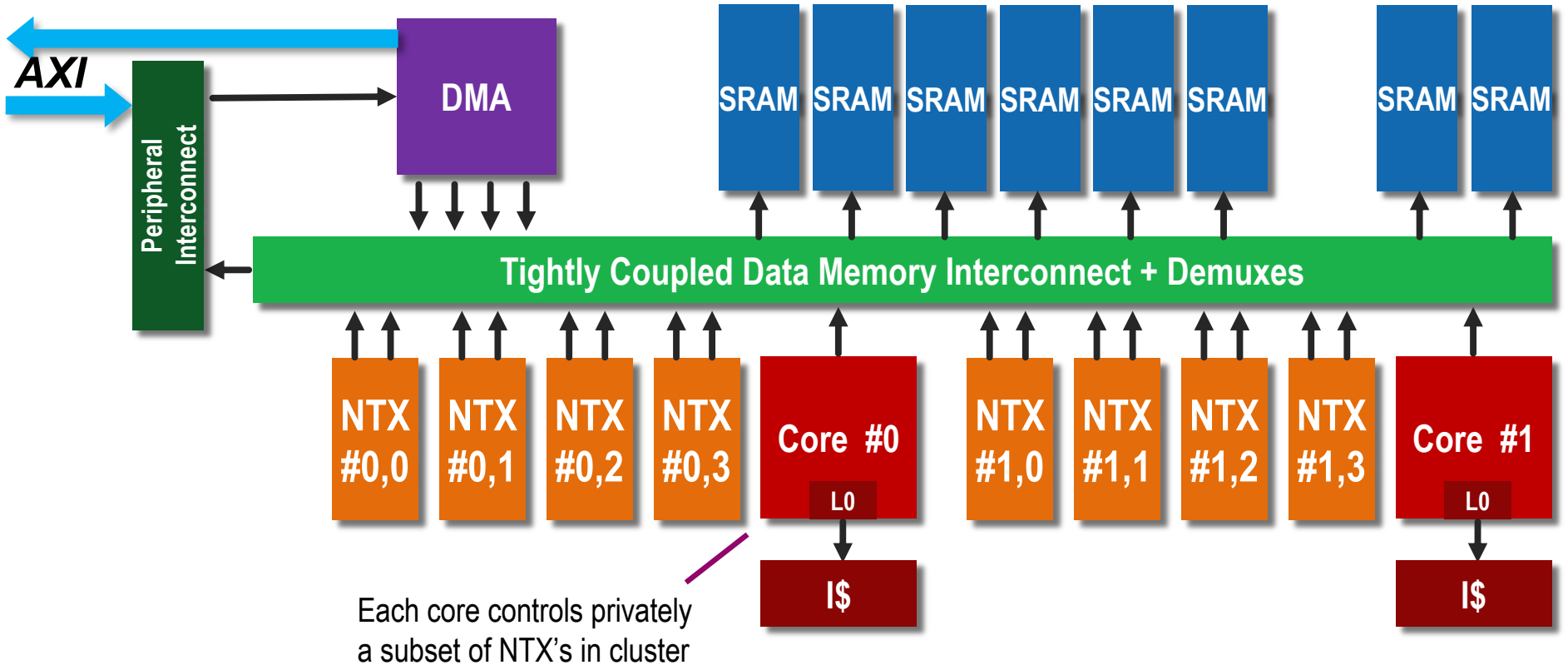
The **Neural Training Accelerator** (**NTX**) [4] is built around a ***float32 fused multiply-accumulate*** core specialized for deep learning applications (ReLU, masking…)



…NTX is ~ the example MAC engine «on steroids»

[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

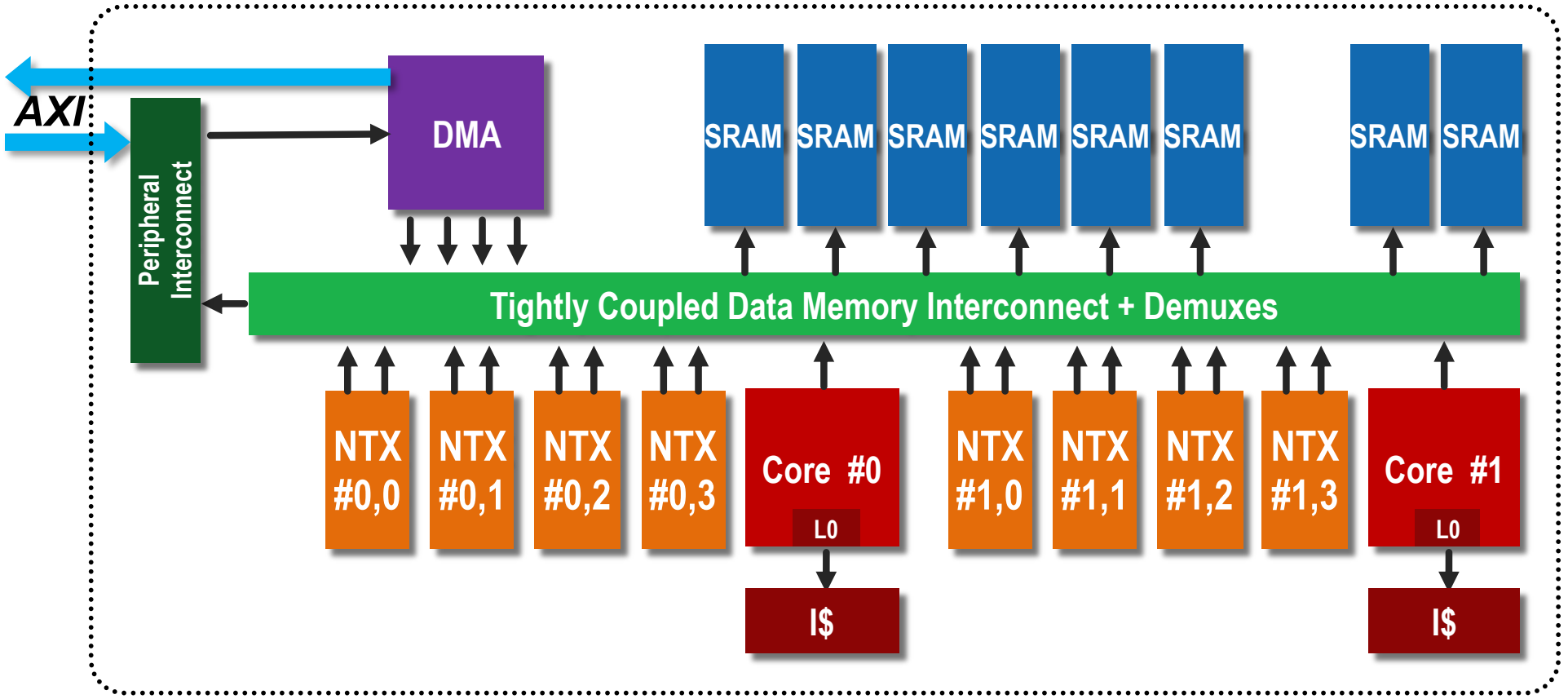
NTX-Augmented Clusters



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

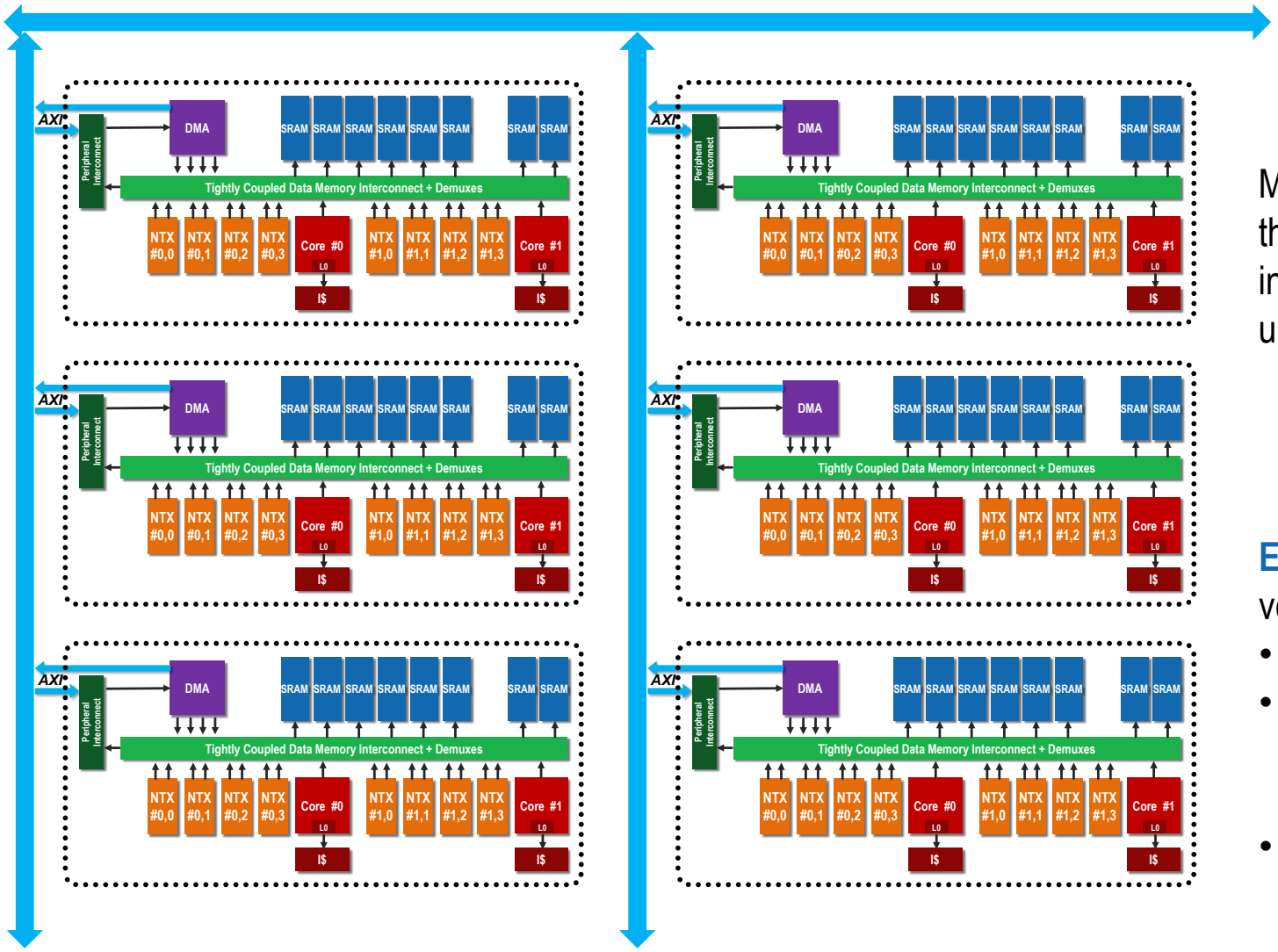
[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

NTX-Augmented Clusters



[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

NTX-Augmented Clusters



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

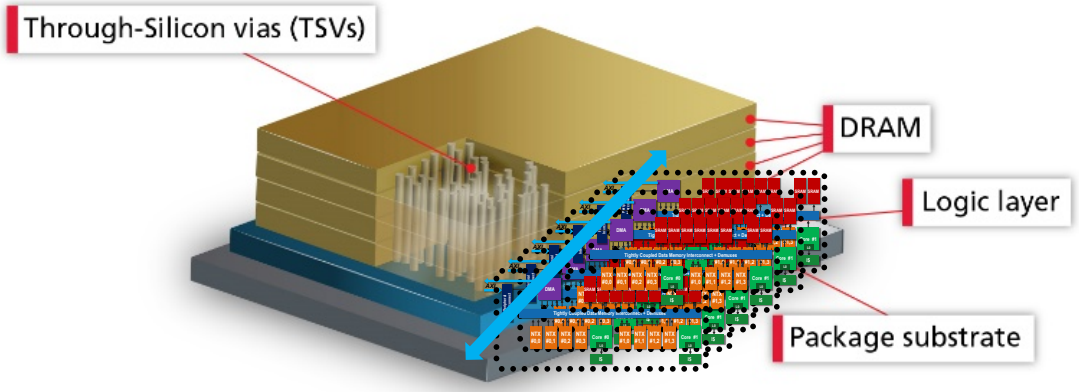
ExaConv platform with early version of NTX:

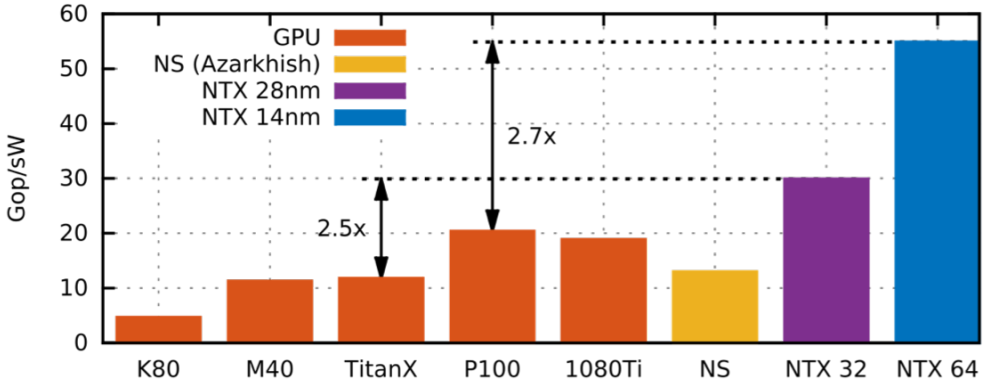
- ST 28nm FD-SOI
- combines **5 clusters** with **2 microcontrollers** + **8 NTX** each
- up to **40 GFlop/s** @500MHz

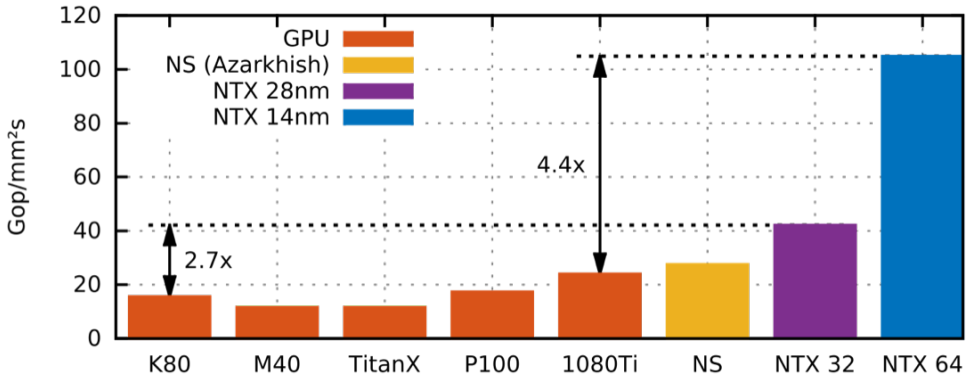
[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

NTX In-Memory Computing: better than GPUs?

Integrating *NTX-augmented clusters* into the logic layer of a Hybrid Memory Cube could lead to **better efficiency** than SoA GPUs at a **lower cost** in terms of Gop/s/mm2 (= Gop/s/$$$)



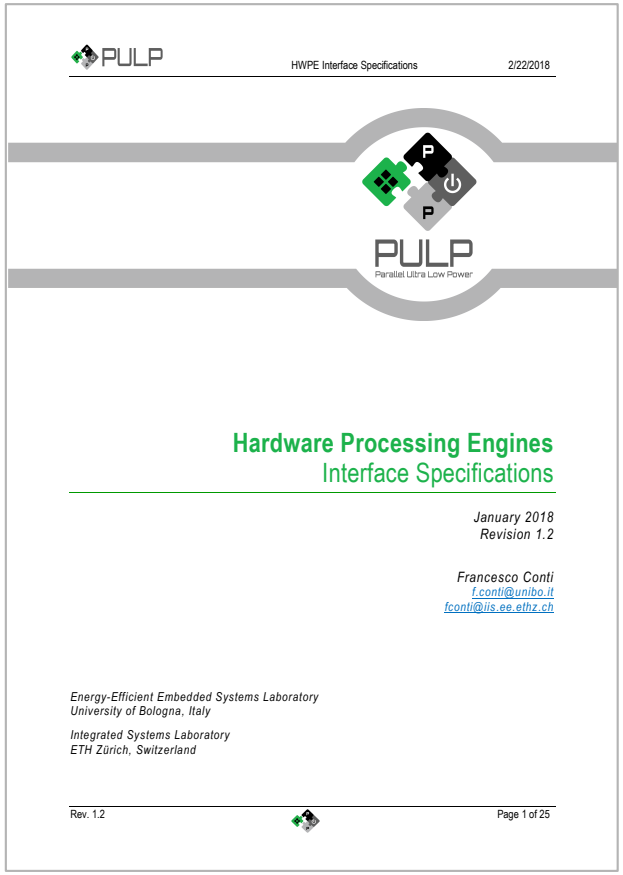




[4] F. Schuiki et al., A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets, under revision

Can *you* build a complex HWPE?

What is available as open-source @ https://github.com/pulp-platform ?


PULP HWPE Interface Specifications 2/22/2018

PULP
Parallel Ultra Low Power

Hardware Processing Engines
Interface Specifications

January 2018
Revision 1.2

Francesco Conti
f.conti@unibo.it
fconti@iis.ee.ethz.ch

Energy-Efficient Embedded Systems Laboratory
University of Bologna, Italy

Integrated Systems Laboratory
ETH Zürich, Switzerland

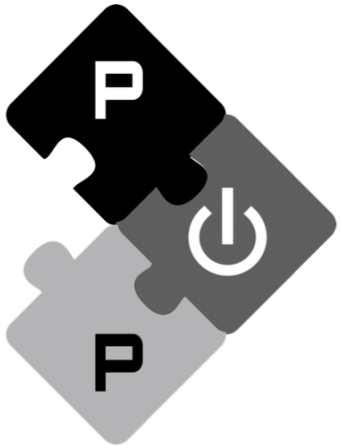
Rev. 1.2 Page 1 of 25

hwpe-stream: a set of «streamer» IPs that can be used to couple stream-based accelerators with shared memories

hwpe-ctrl: IPs to implement a memory-mapped register file, plus a microcode processor to implement arbitrary HW loops

documentation and interface protocol specifications

Can *you* build a complex HWPE?



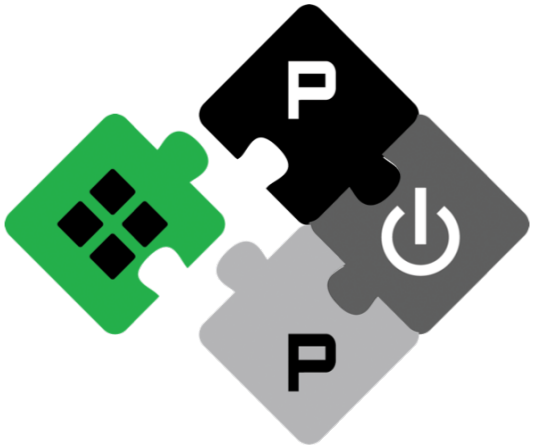
Can *you* build a complex HWPE?

Find all necessary information in the **PULPissimo** repository:

https://github.com/pulp-platform/pulpissimo

your datapath, optimized

for your application



Special thanks to: **Fabian Schuiki** and **Michael Schaffner** for the NTX design, **Robert Schilling** (IAIK – TU Graz) for the HWCrypt design, **Davide Schiavone** for his help on HWCE.

Thanks for your attention. Questions?