RISC-V® Summit

December 8-10 | Virtual Event

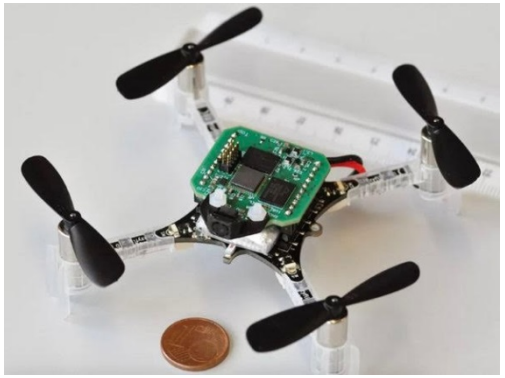
An Open Flow for DNNs on Ultra-Low-Power RISC-V Cores

Dr. Francesco Conti
Assistant Professor
University of Bologna

#RISCVSUMMIT

Real-World DNNs at Extreme-Edge?

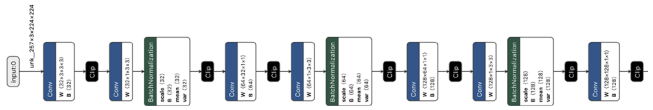
improve latency



preserve privacy

enhance energy efficiency

Real-World DNNs at Extreme-Edge?



specification & dataset selection

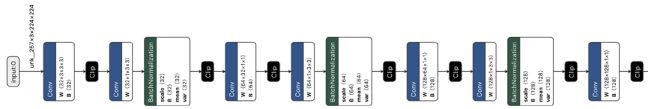
training

?

optimized HW & architecture

How to bridge the **algorithm** <-> **device** divide?

Real-World DNNs at Extreme-Edge?



specification & dataset selection

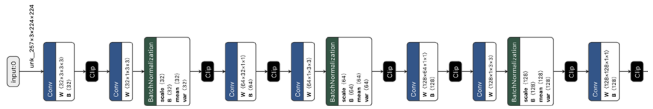
training

optimized DNN primitives

optimized HW & architecture

1. Exploit hardware's strengths

Real-World DNNs at Extreme-Edge?



specification & dataset selection

training

graph optimization

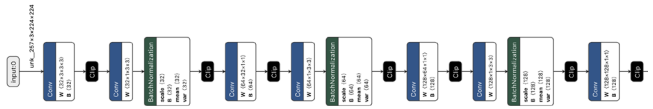
memory-aware deployment

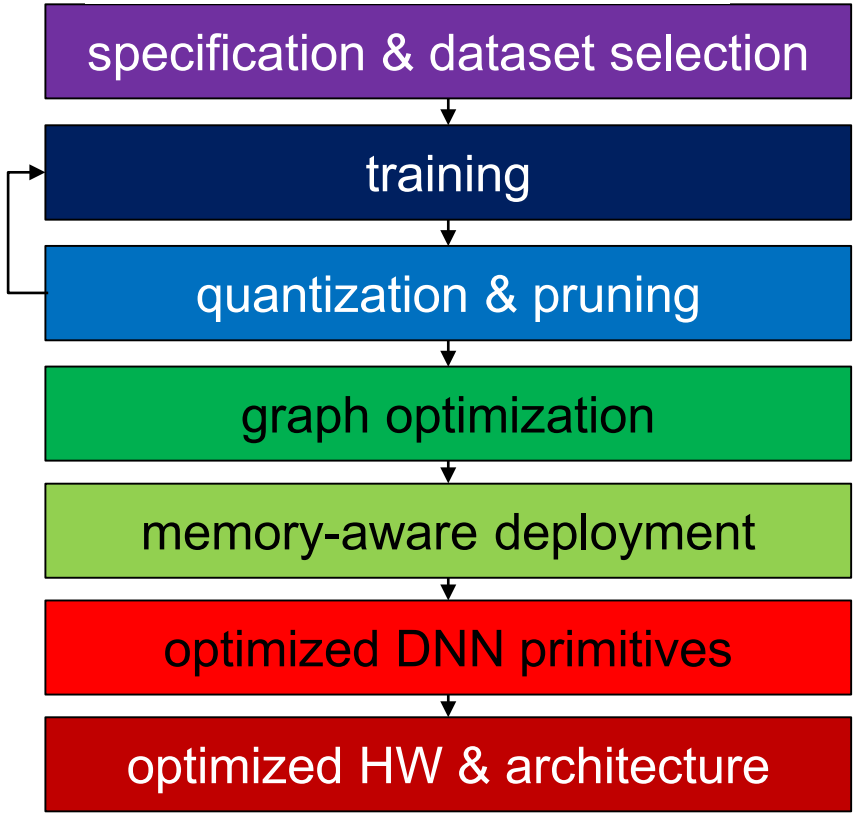
optimized DNN primitives

optimized HW & architecture

2. Keep compute units fed with data
from on-chip memories; minimize off-chip access

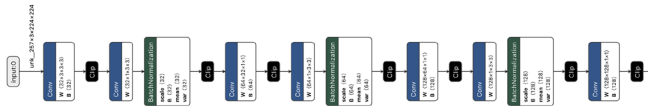
Real-World DNNs at Extreme-Edge?





3. Produce DNNs that are tuned to embedded hardware (e.g., **int8**)

Real-World DNNs at Extreme-Edge?



specification & dataset selection

training

quantization & pruning

graph optimization

memory-aware deployment

optimized DNN primitives

optimized HW & architecture

Open-Source flow for DNN Deployment on PULP devices

NEMO
NE*ural* ***M****inimization*
*for pyt****O****rch*

DORY
D*eployment* ***O****riented to memo****RY***

PULP-NN
P*arallel* ***ULP***
N*eural* ***N****etwork library*

Real-World DNNs at Extreme-Edge?

specification & dataset selection

training

quantization & pruning

graph optimization

memory-aware deployment

optimized DNN primitives

optimized HW & architecture

THIS TALK

NEMO
NEural Minimization for pytOrch

DORY
Deployment Oriented to memoRY

PULP-NN
Parallel ULP Neural Network library

PULP-NN: optimized back-end

specification & dataset selection

training

quantization & pruning

graph optimization

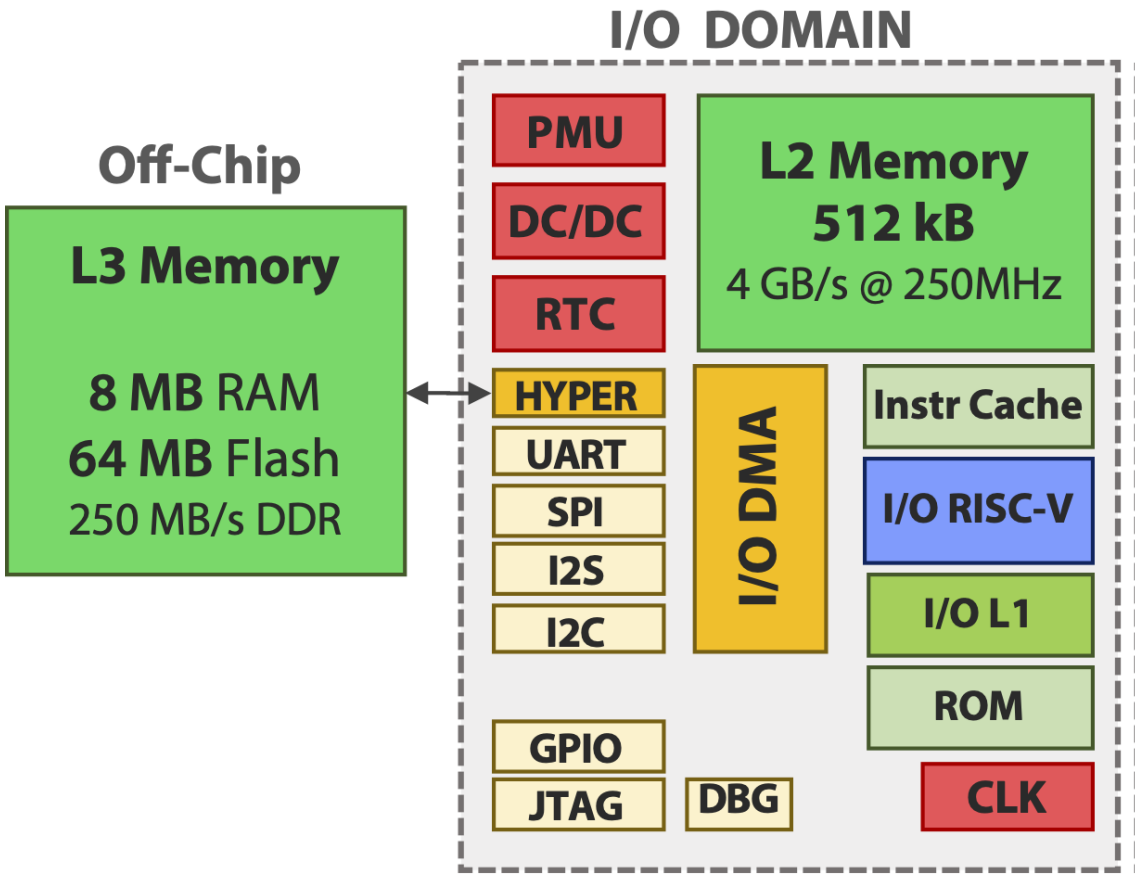
memory-aware deployment

optimized DNN primitives

optimized HW & architecture

PULP-NN
P**arallel **ULP
***N**eural **N**etwork library*

GreenWaves GAP8 Architecture



GreenWaves GAP8 is a **RISC-V** ultra-low power processor based on the open-source **PULP** (Parallel Ultra-Low-Power) paradigm

- a fast microcontroller with autonomous I/O capability, support for off-chip memory (HyperRAM DRAM / Flash)
- RV32IMC in-order, 4 pipeline stage core (RI5CY, now OpenHW Group CV32E40P)

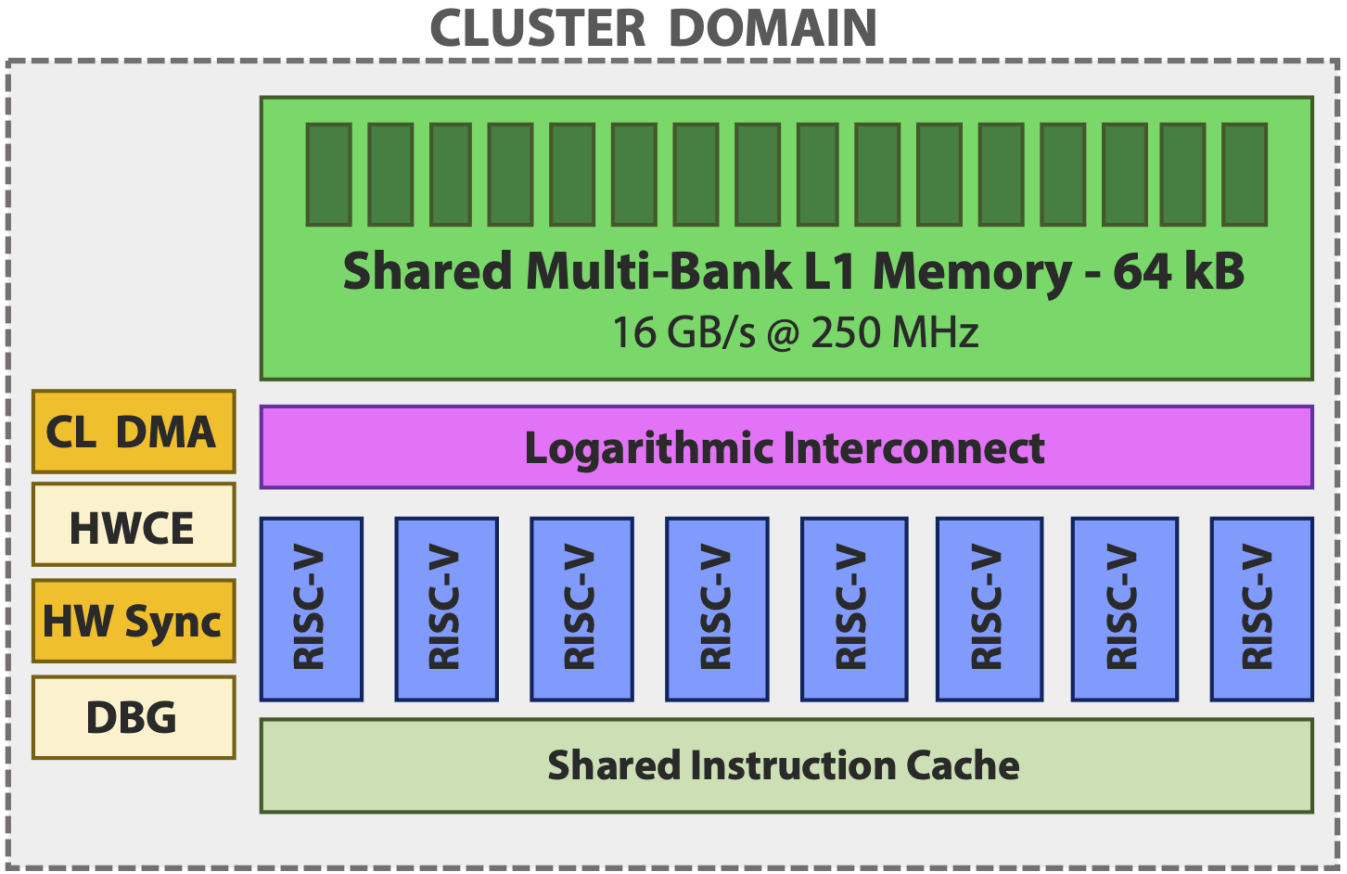
GreenWaves GAP8 Architecture

GreenWaves GAP8 is a **RISC-V** ultra-low power processor based on the open-source **PULP** (Parallel Ultra-Low-Power) paradigm

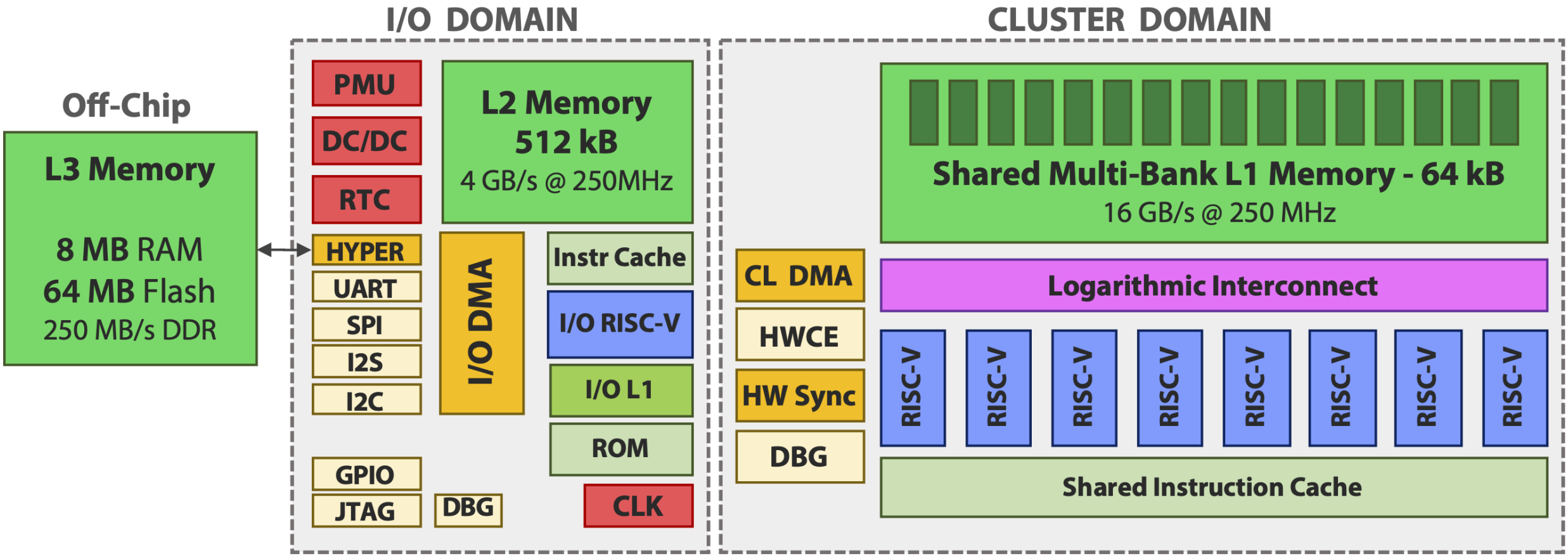
- a programmable **cluster** of **RI5CY cores** sharing memory at L1

- high-bandwidth, low-latency L1 scratchpad + **DMA** for manual memory management

- **xPULPv2** ISA extension for enhanced DSP capabilities: hardware loops, address post-increment, SIMD scalar product



GreenWaves GAP8 Architecture



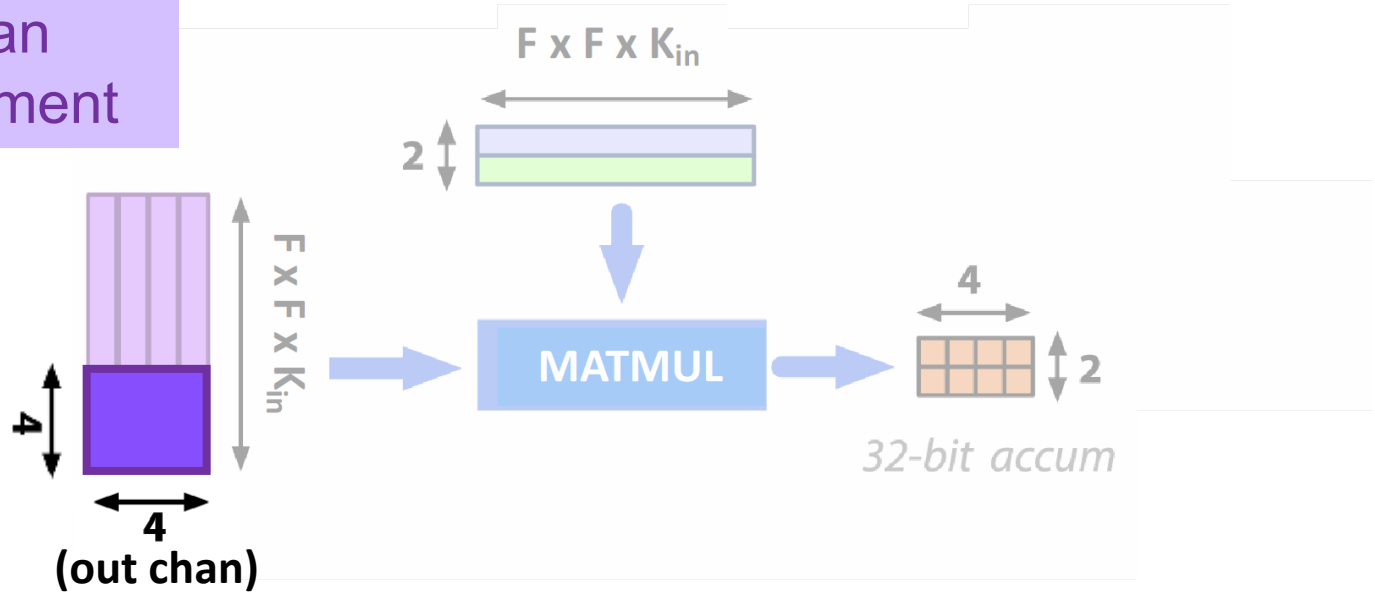
PULP-NN: optimized computational back-end

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Load 16 weights (8-bit)
4 out chan, 4 in chan
address post-increment



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

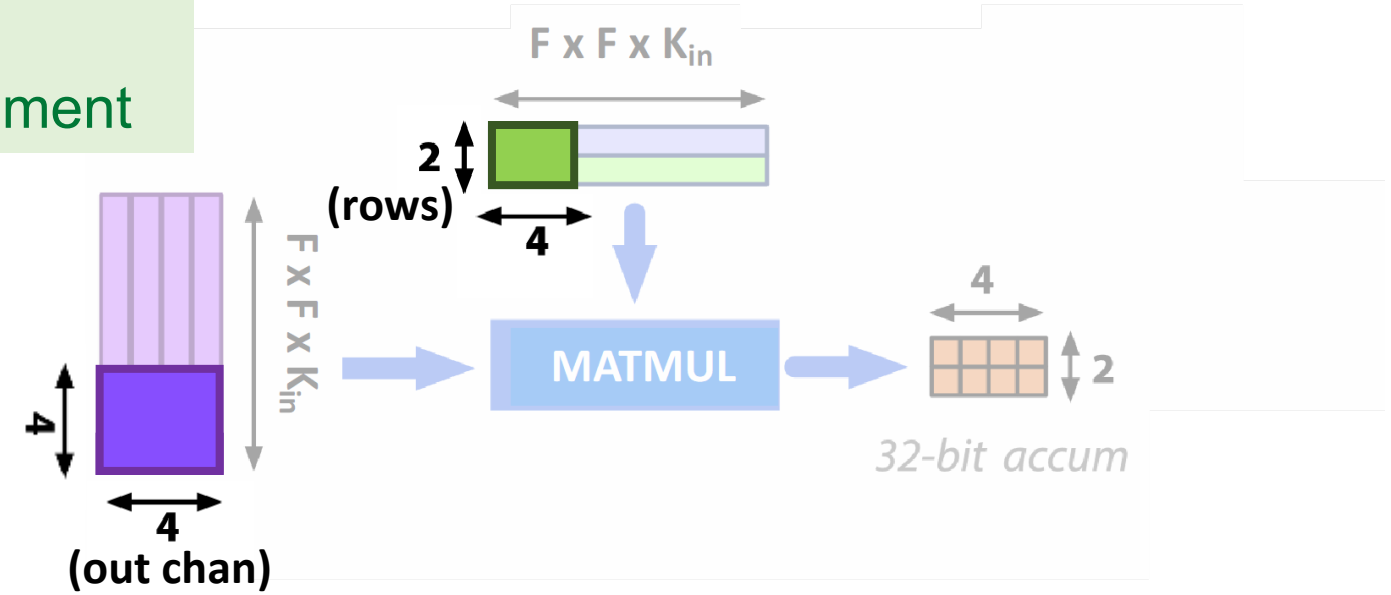
PULP-NN: optimized computational back-end

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Load 8 pixels
2 rows, 4 in chan
address post-increment



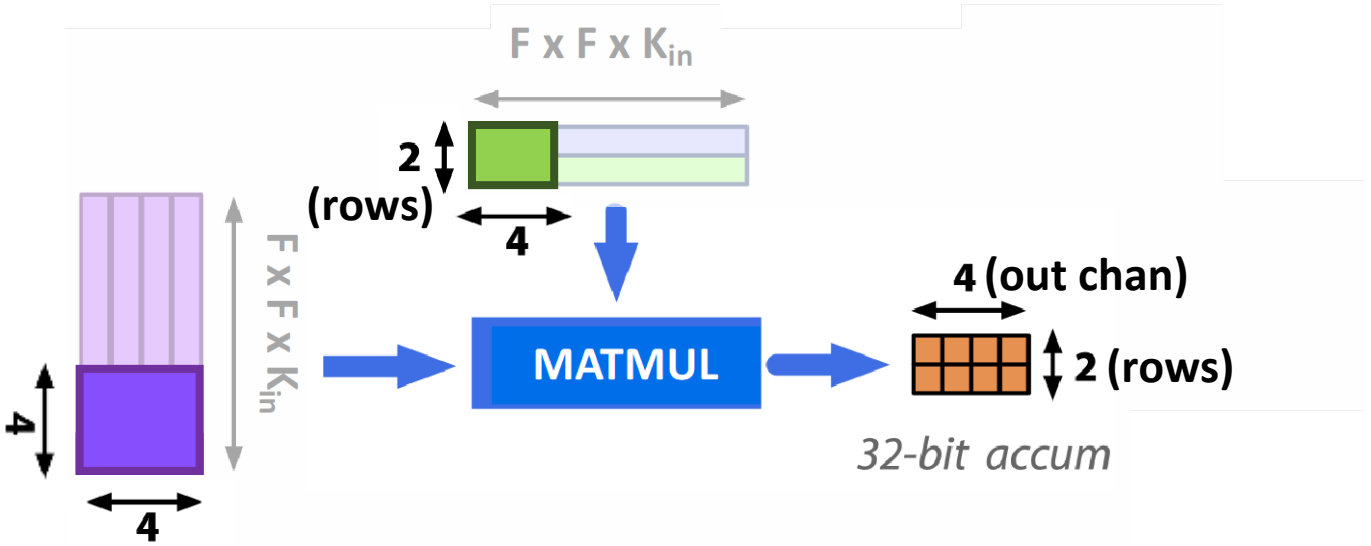
PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

PULP-NN: optimized computational back-end

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```



Compute 32 MAC over 8 accumulators
dot-product instructions

PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

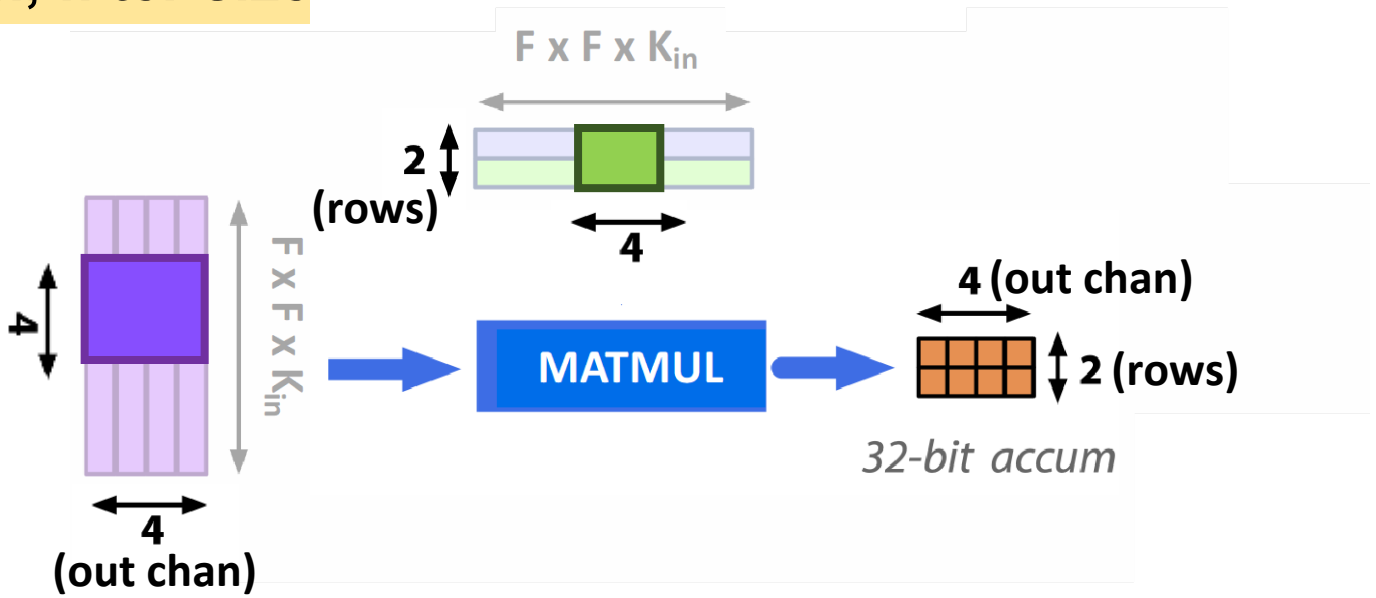
PULP-NN: optimized computational back-end

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Loop over in chan, filter size



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

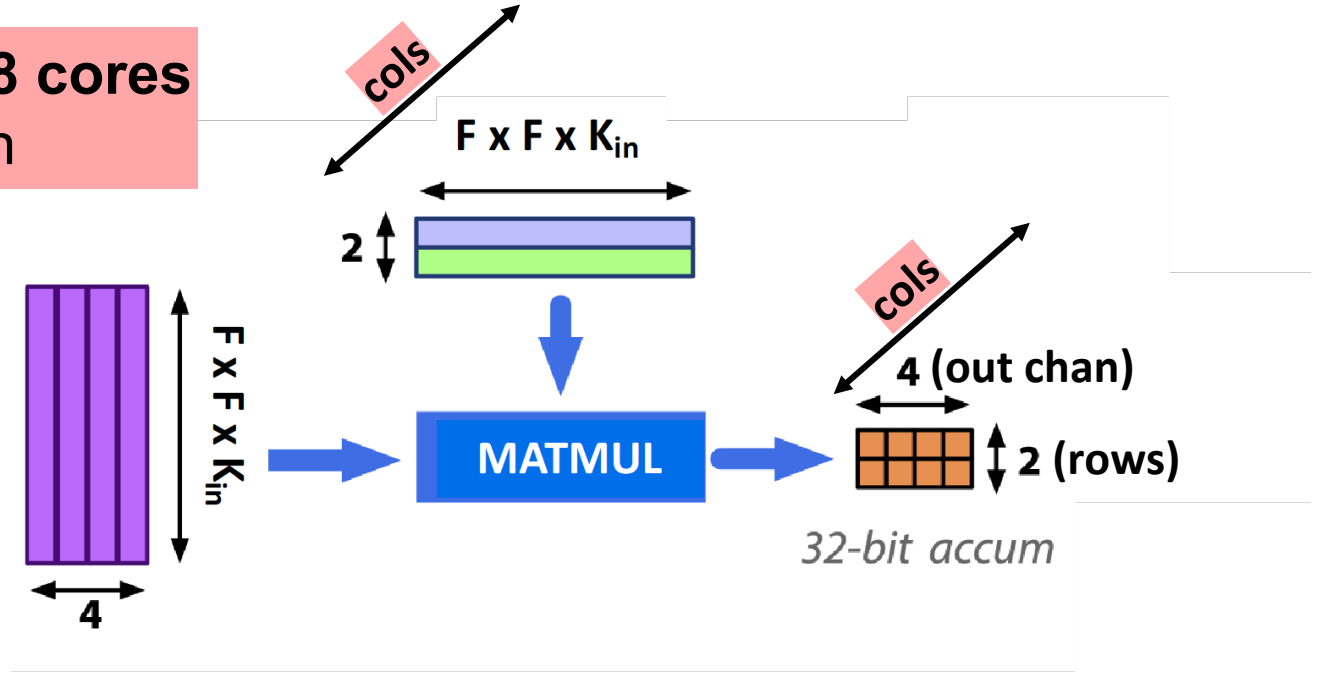
PULP-NN: optimized computational back-end

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Parallelize over 8 cores
column dimension



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

PULP-NN: optimized computational back-end

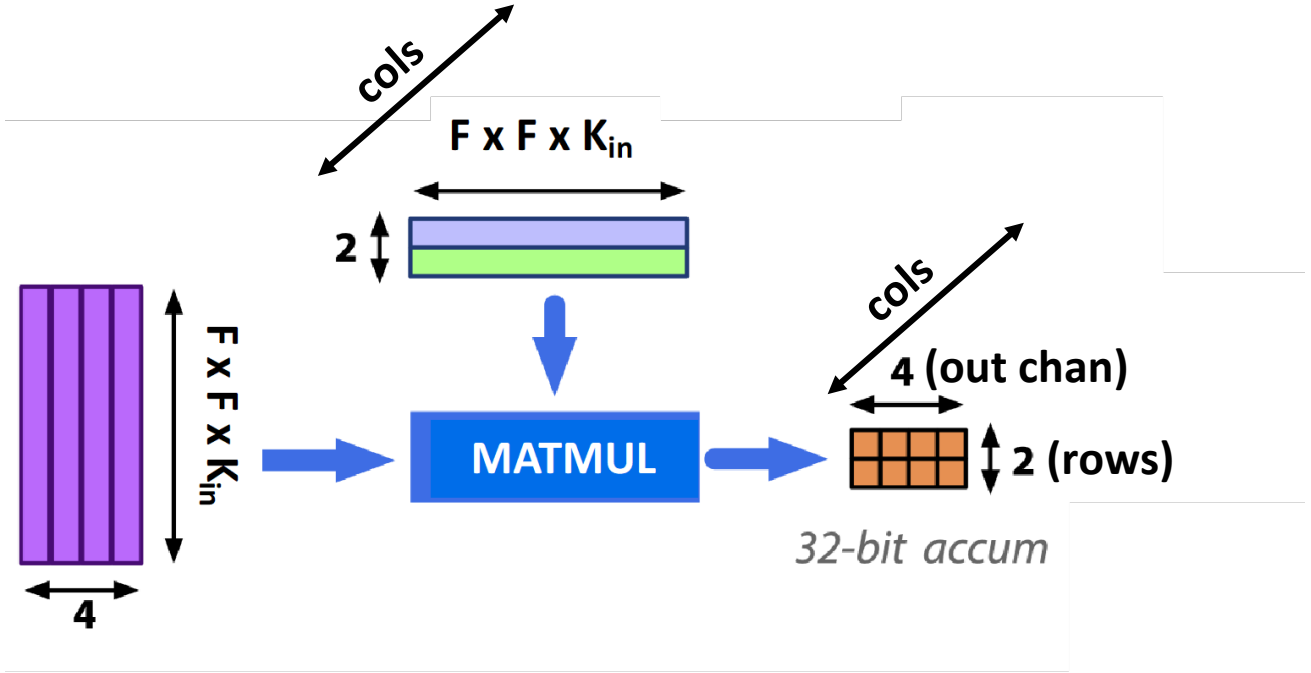
Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

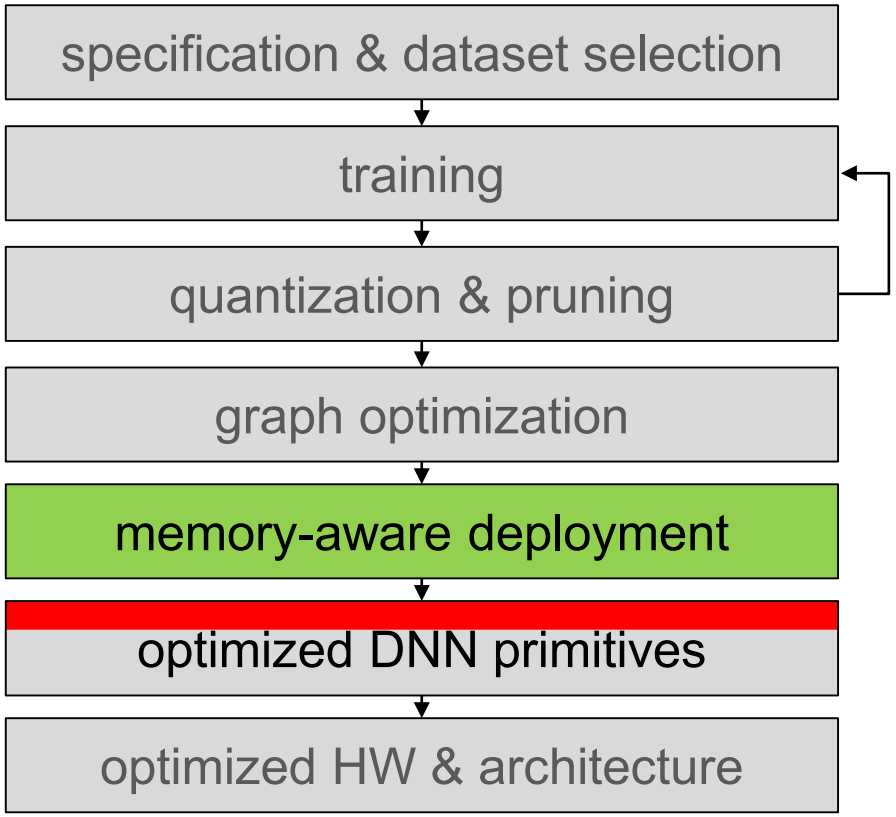
Peak Performance (8 cores)

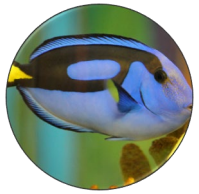
15.5 MAC/cycle



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

DORY: Tiling & Code Generation

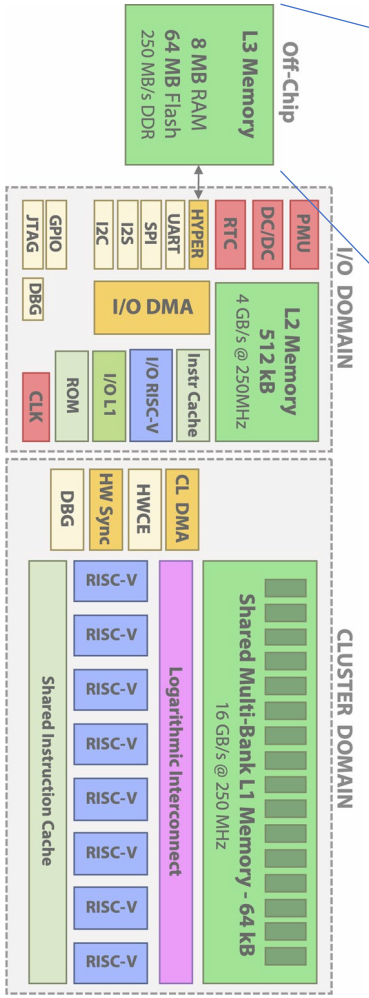




DORY

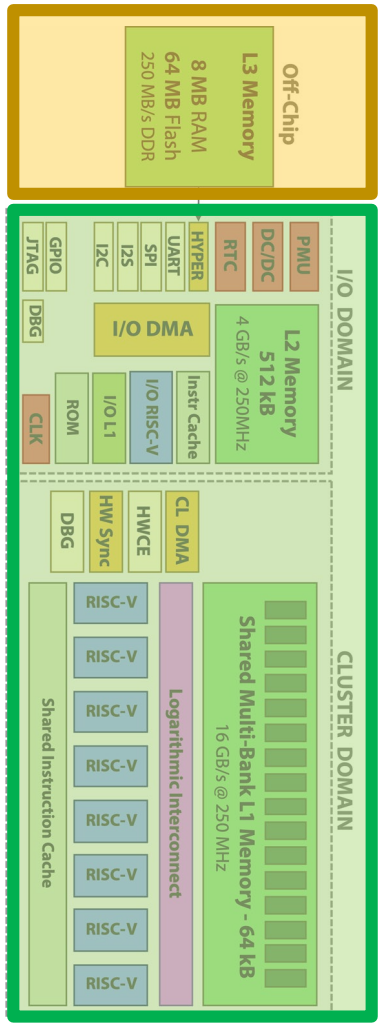
D*eployment* ***O****riented to memo****RY***

DORY: the tensor tiling problem



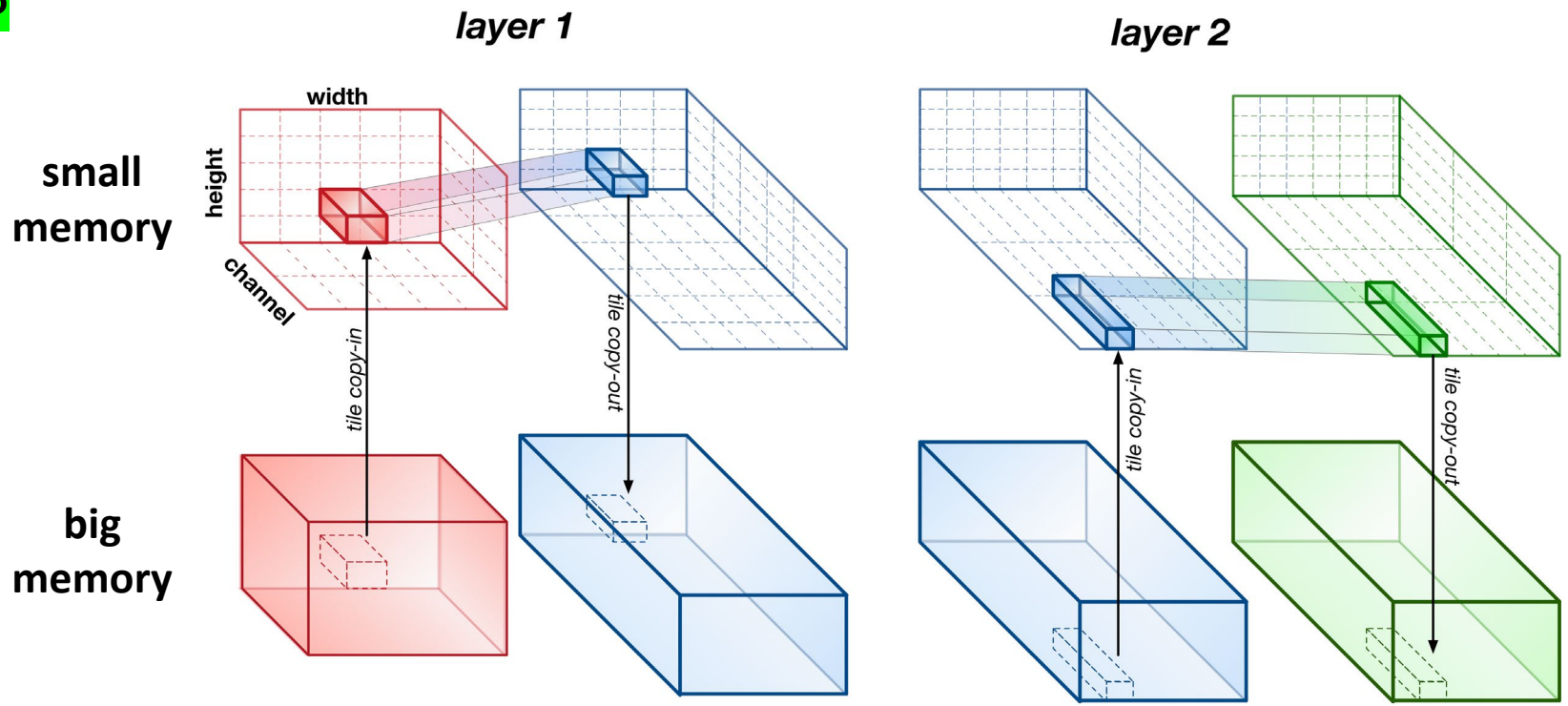
1.0-MobileNet-128
(59% top-1 accuracy on ImageNet)

DORY: the tensor tiling problem

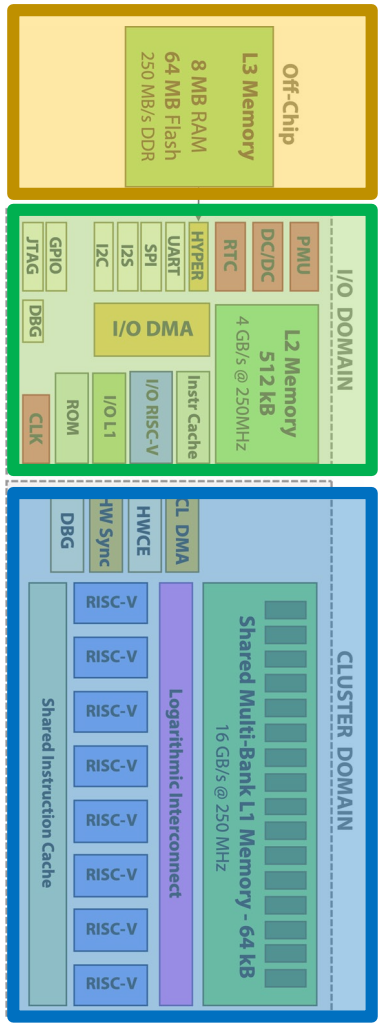


L3 / L2 tiling

64 MB / 512 kB

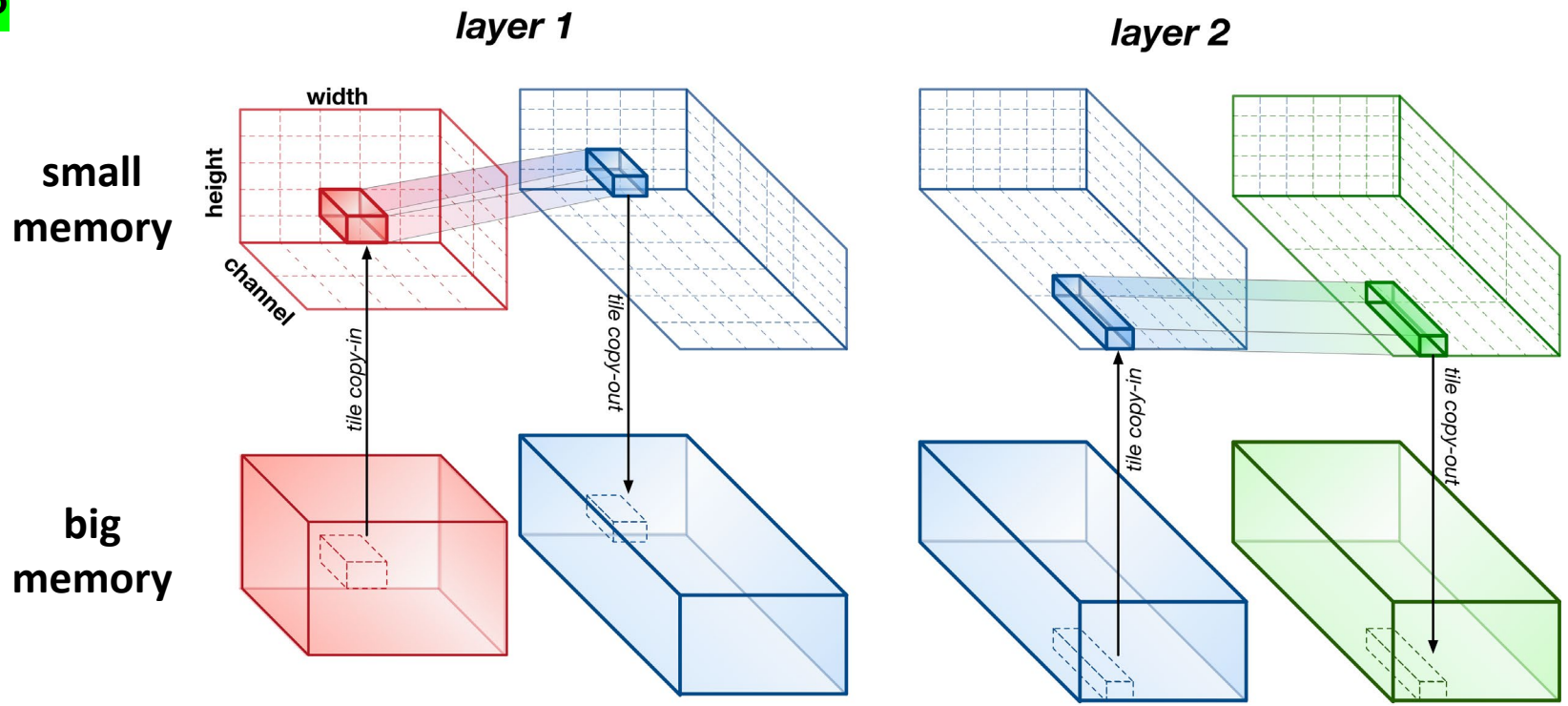


DORY: the tensor tiling problem



L3 / L2 tiling
64 MB / 512 kB

L2 / L1 tiling
512 kB / 64 kB



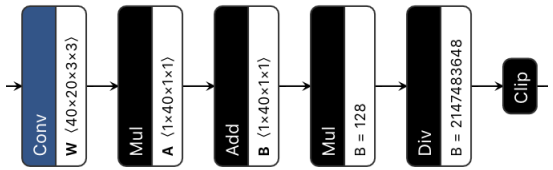
DORY: Tiling as optimization problem

How to define the sizes of various tiles? → abstracted as **Constraint Programming** problem

$$\text{cost} = \mathbf{max}\ Size(\boldsymbol{W_{tile}}) + Size(\boldsymbol{x_{tile}}) + Size(\boldsymbol{y_{tile}})$$

MEMORY → $\text{s.t.} \quad Size(\boldsymbol{W_{tile}}) + Size(\boldsymbol{x_{tile}}) + Size(\boldsymbol{y_{tile}}) < SizeSmallMem$

Basic idea: maximize tile size while fitting the overall memory size constraint



Integer DNN

Google **ORTools**

Integer DNN + tile sizes

DORY: Tiling as optimization problem

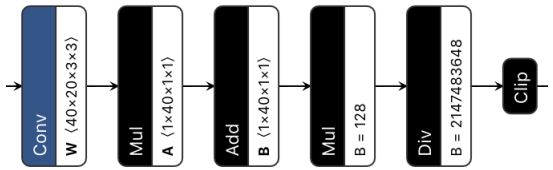
How to define the sizes of various tiles? → abstracted as **Constraint Programming** problem

$$\text{cost} = \max Size(\boldsymbol{W_{tile}}) + Size(\boldsymbol{x_{tile}}) + Size(\boldsymbol{y_{tile}})$$

MEMORY → s. t. $Size(\boldsymbol{W_{tile}}) + Size(\boldsymbol{x_{tile}}) + Size(\boldsymbol{y_{tile}}) < SizeSmallMem$

GEOMETRY → s. t. $\{\boldsymbol{y_{tile}}[ch_{out}] = \boldsymbol{W_{tile}}[ch_{out}], \dots\}$

Size of tiles for in tensors / out tensors / weights tied by geometric constraints typical of a given layer



Integer DNN

Google **ORTools**

Integer DNN + tile sizes

DORY: Tiling as optimization problem

How to define the sizes of various tiles? → abstracted as **Constraint Programming** problem

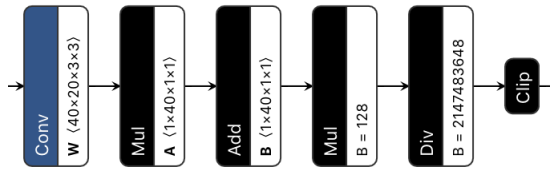
$$\text{cost} = \mathbf{max}\ Size(\boldsymbol{W_{tile}}) + Size(\boldsymbol{x_{tile}}) + Size(\boldsymbol{y_{tile}})$$

MEMORY → $\text{s.t.} \quad Size(\boldsymbol{W_{tile}}) + Size(\boldsymbol{x_{tile}}) + Size(\boldsymbol{y_{tile}}) < SizeSmallMem$

GEOMETRY → $\text{s.t.} \quad \{\boldsymbol{y_{tile}}[ch_{out}] = \boldsymbol{W_{tile}}[ch_{out}], \dots\}$

EFF. HEURISTICS → $\text{cost}' = \text{cost} + \{\boldsymbol{y_{tile}}[ch_{out}]\ \text{divisible by } 4, \dots\}$

Performance is maximum for configurations that use PULP-NN primitives more efficiently (e.g., full parallelism)

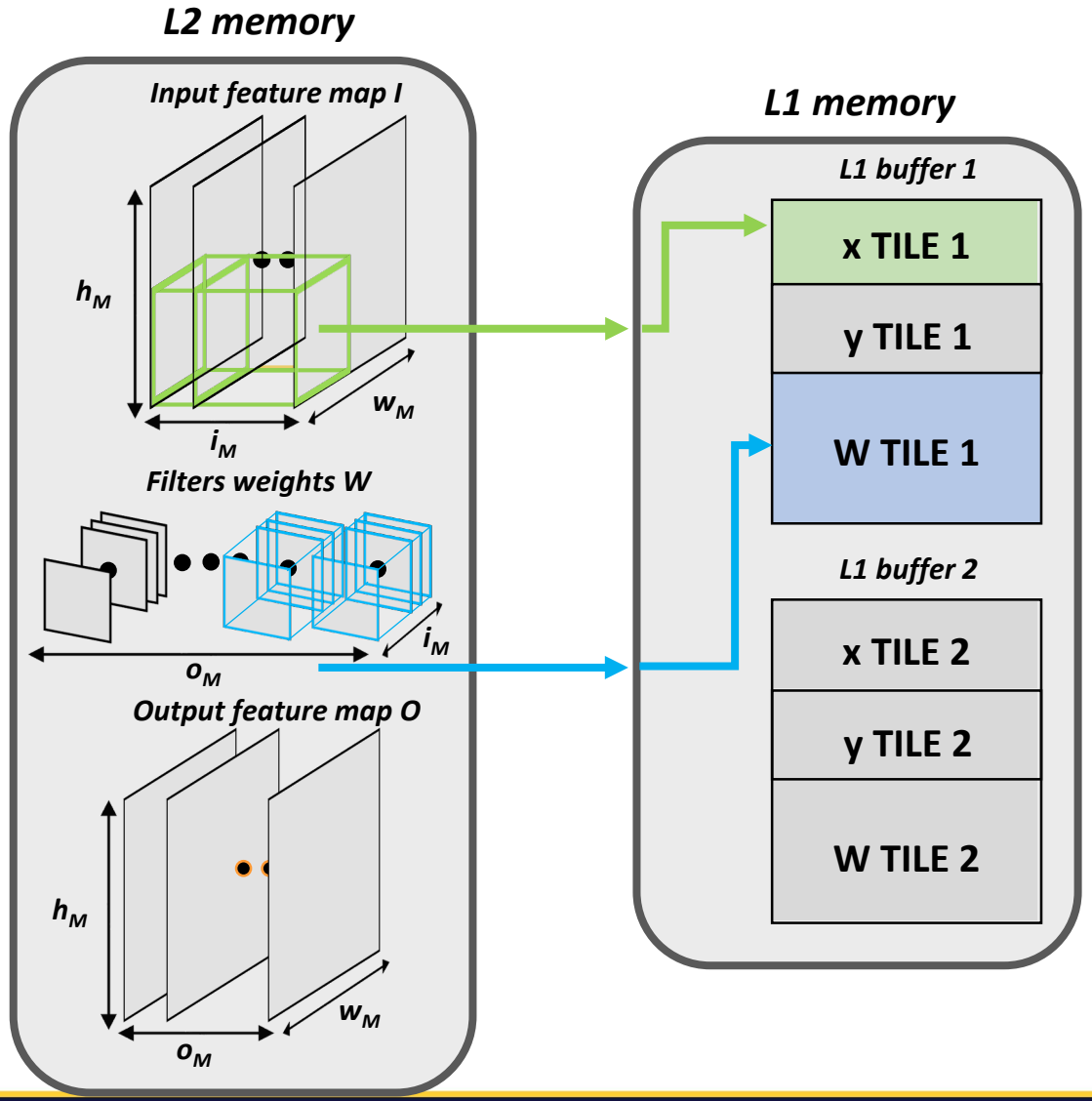


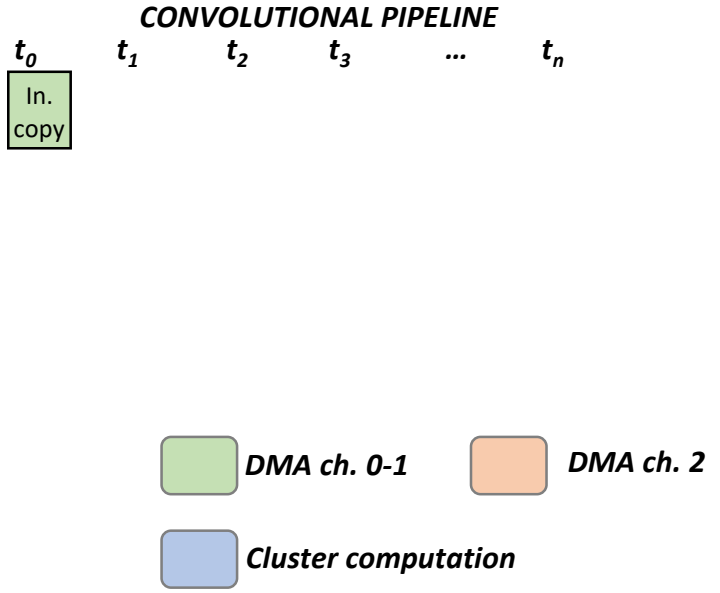
Integer DNN

Google **ORTools**

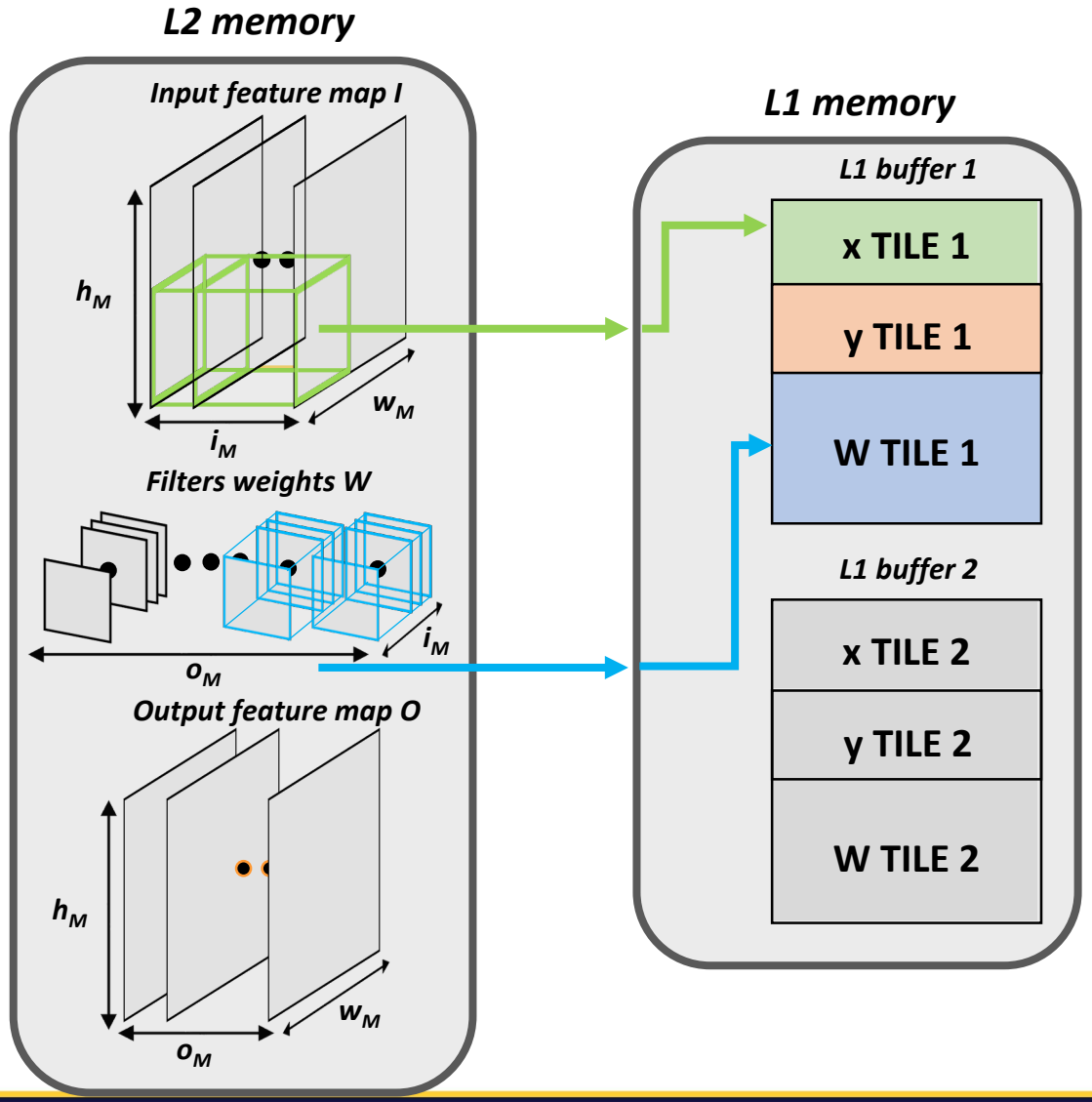
Integer DNN + tile sizes

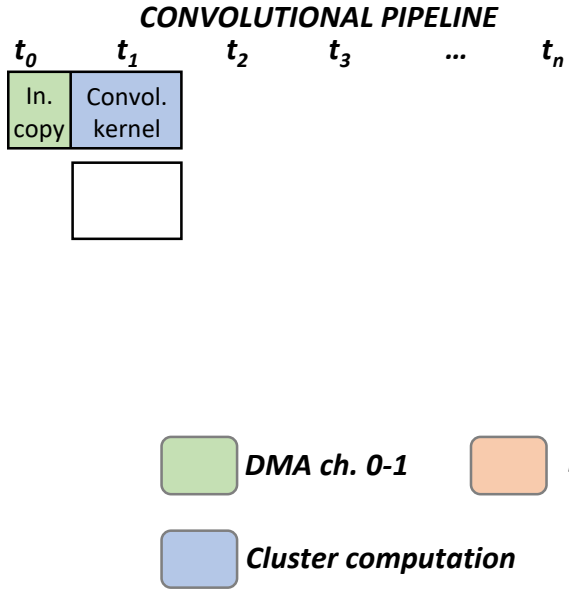
DORY: Tile data movement



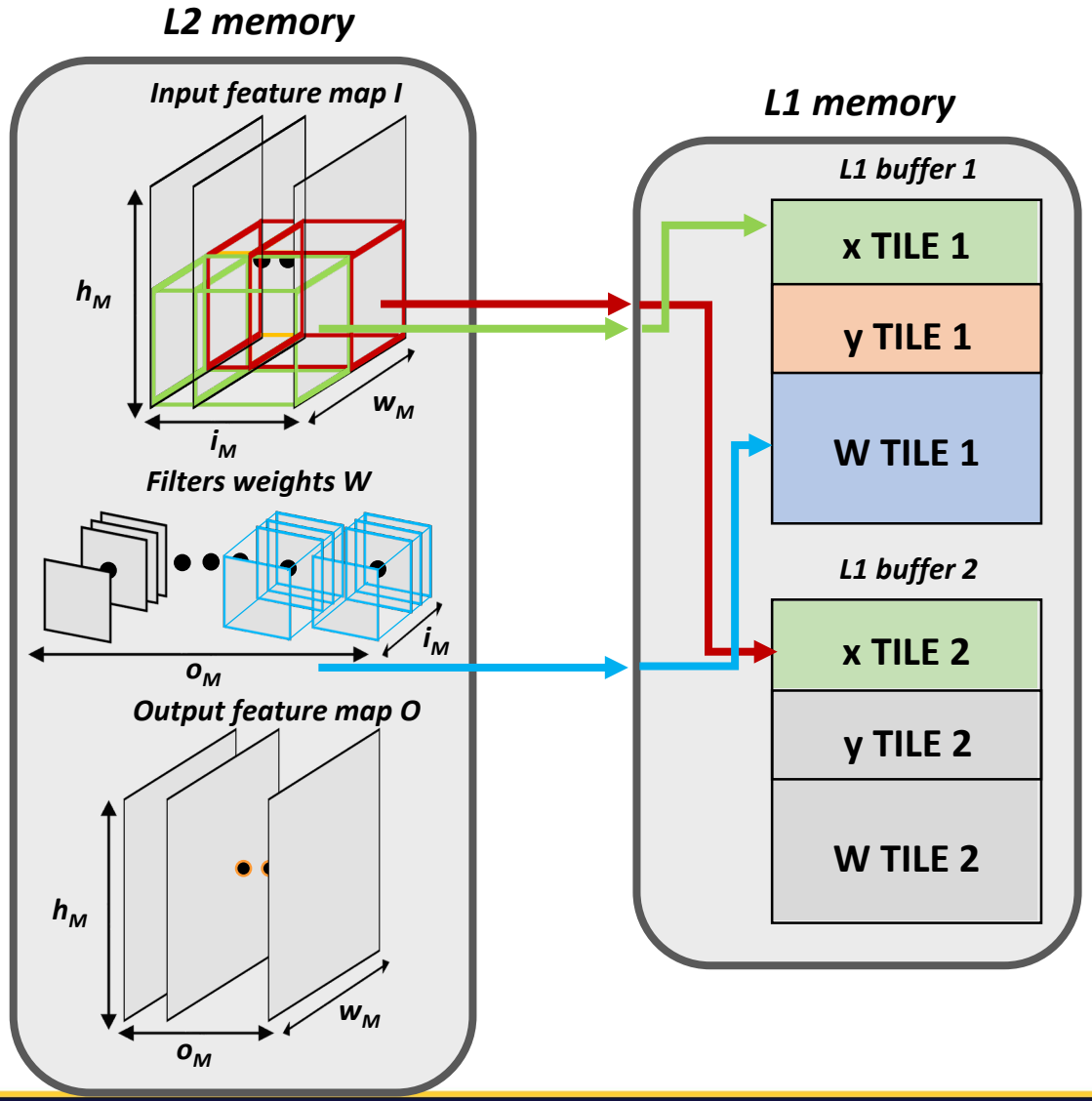


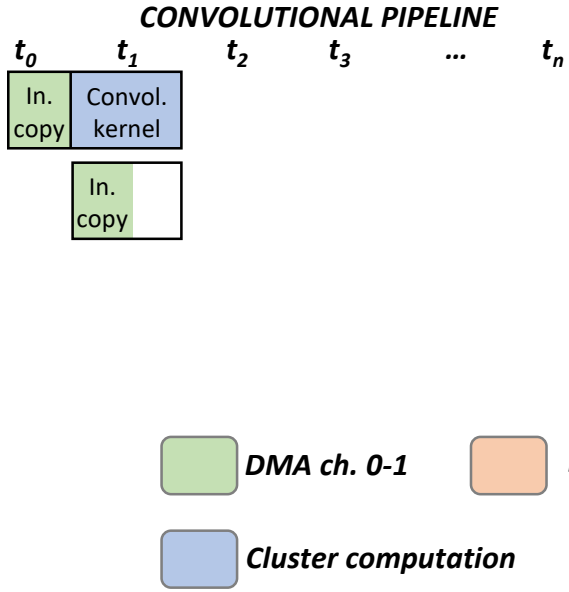
DORY: Tile data movement



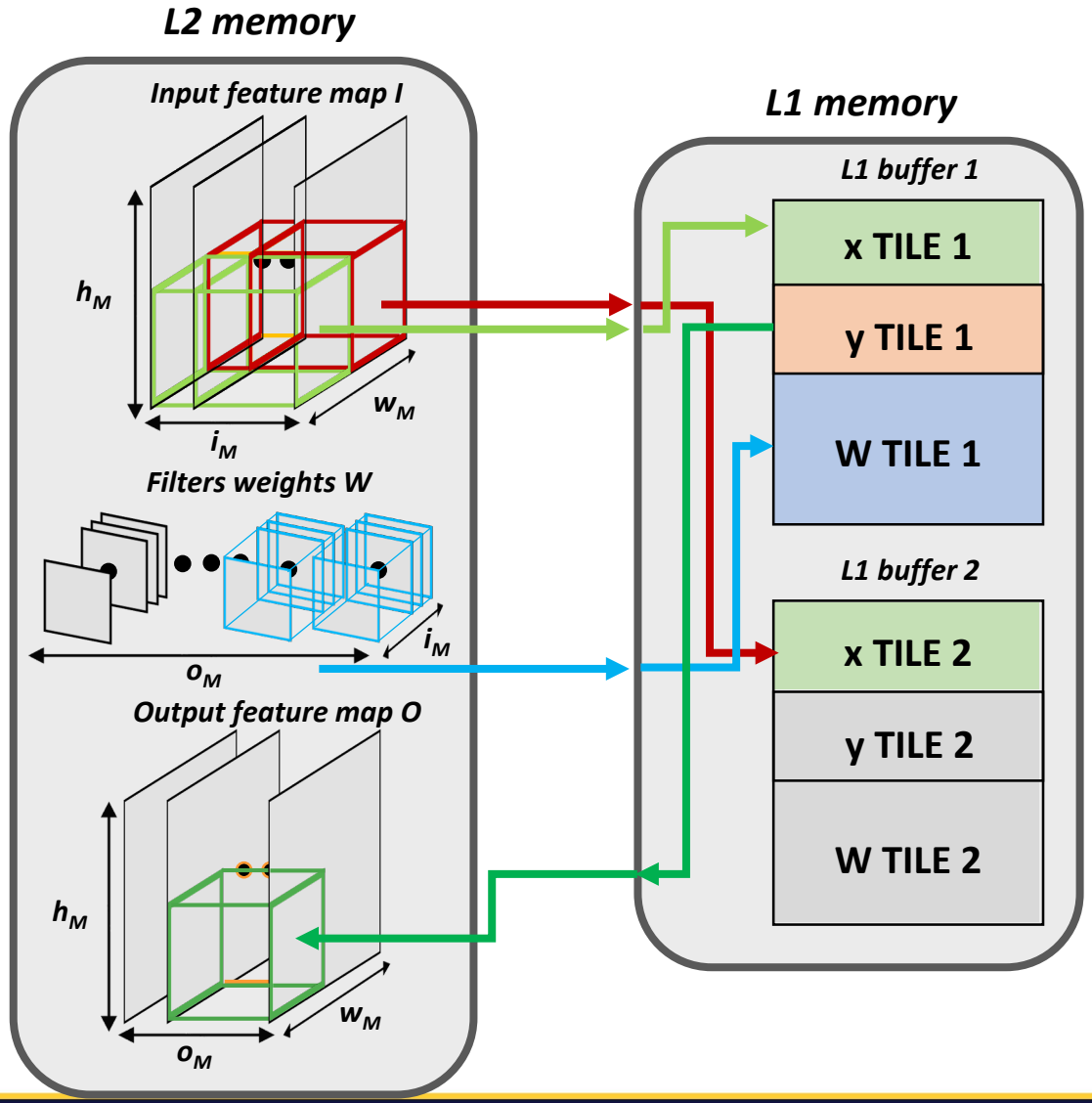


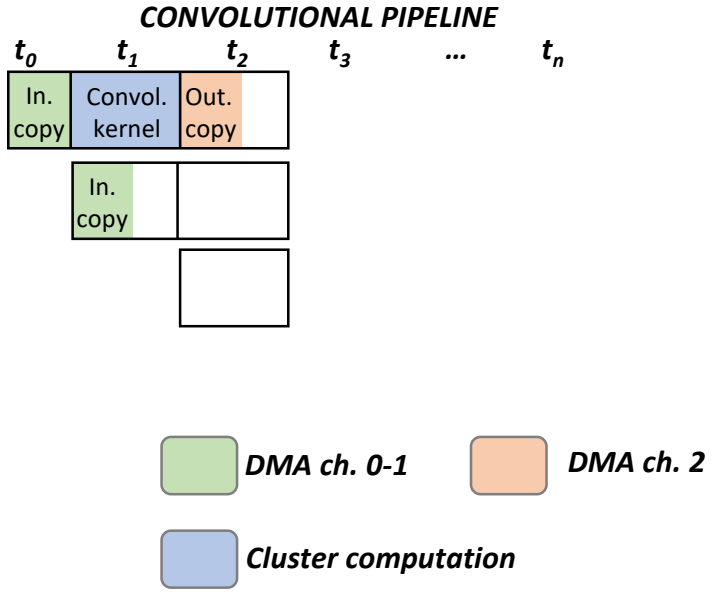
DORY: Tile data movement



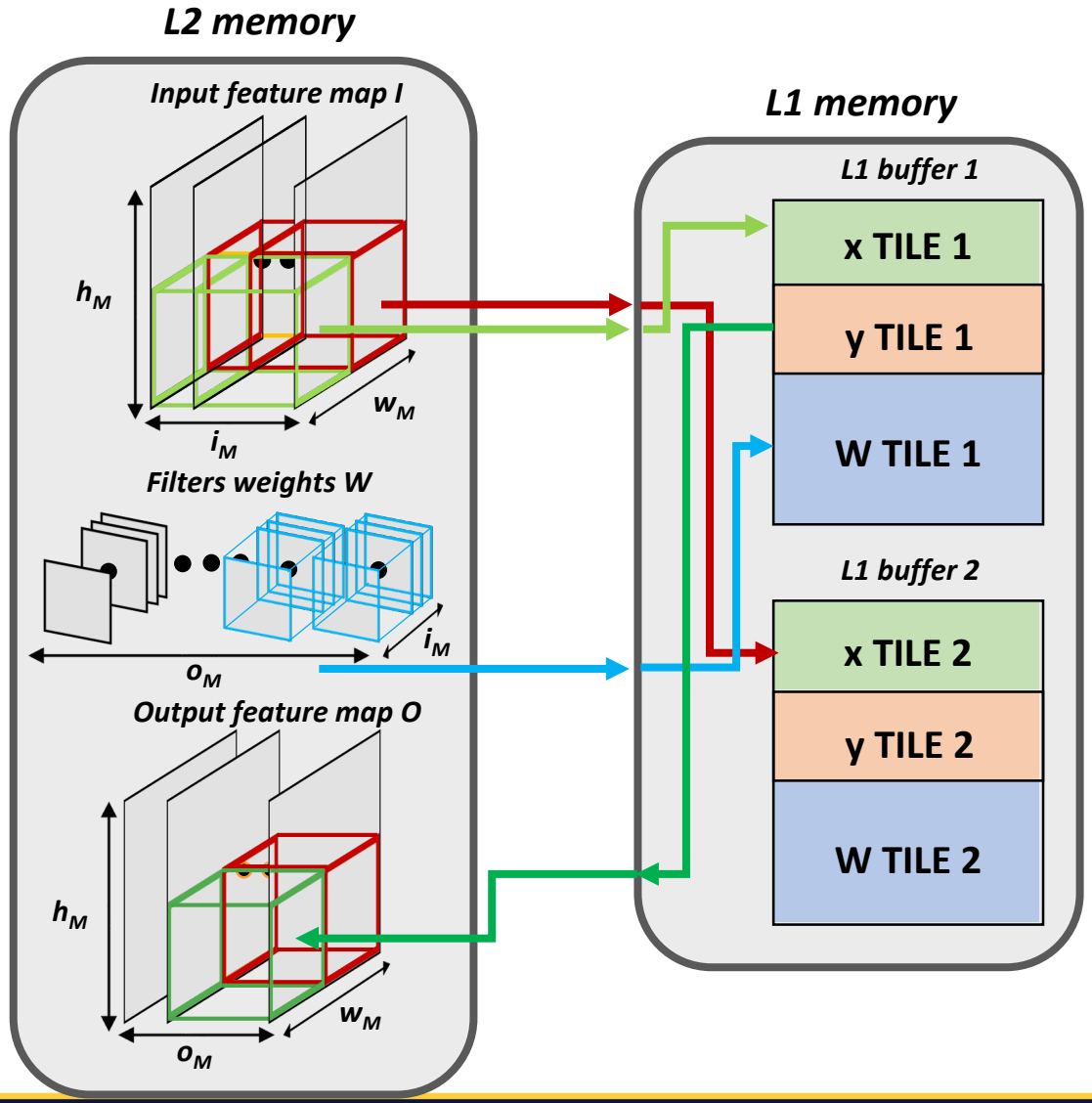


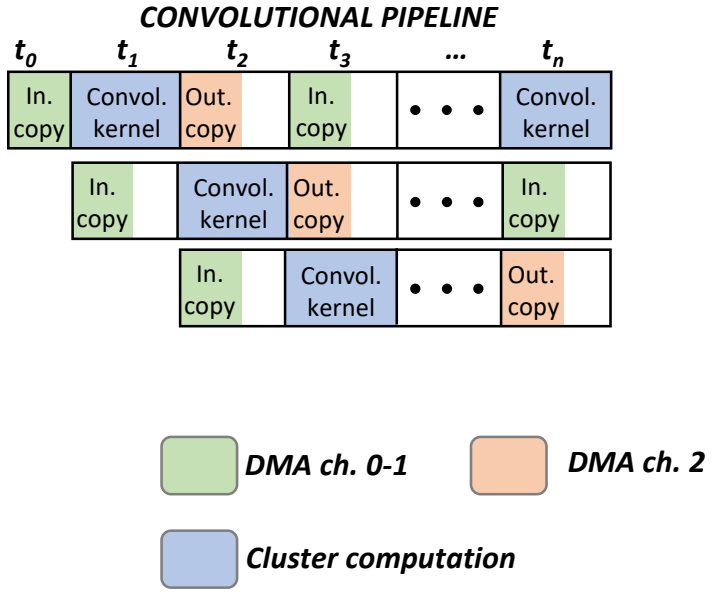
DORY: Tile data movement



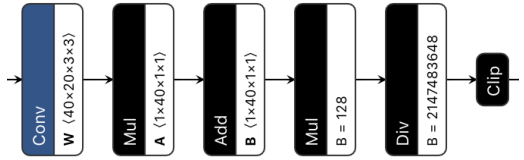


DORY: Tile data movement





DORY: code generation example



Integer Network + tile sizes

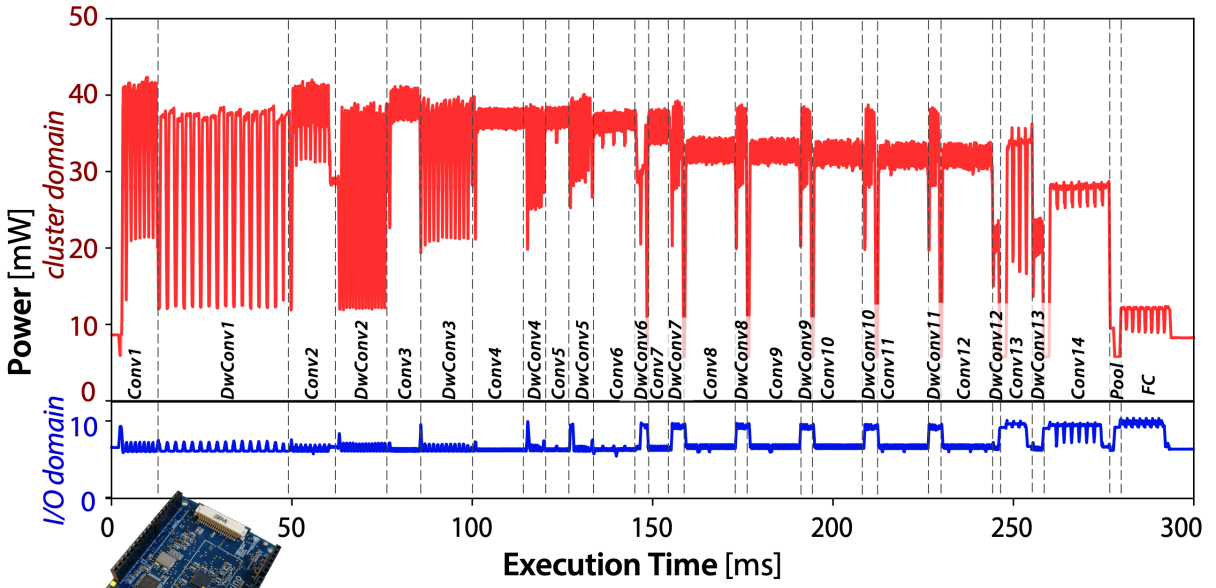
Code Generation
from templates

Network-level C code

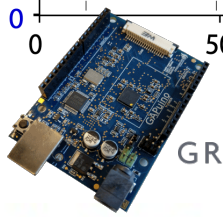
- L3/L2 transfer boilerplate
- double buffering for weights
- calls to layer-level code

Layer-level C code

- L2/L1 transfer boilerplate
- calls to PULP-NN backend library



MMAC	Avg Pwr [mW]	Time [ms]	
3.54	42	9.75	Conv1
1.18	37	36.92	DwConv1
8.39	42	13.05	Conv2
0.59	33	13.95	DwConv2
8.39	44	8.44	Conv3
1.18	40	15.08	DwConv3
16.78	43	14.25	Conv4
0.29	37	4.59	DwConv4
8.39	43	7.19	Conv5
0.59	42	6.06	DwConv5
16.78	42	13.30	Conv6
0.15	39	1.56	DwConv6
8.39	40	7.03	Conv7
0.29	42	3.03	DwConv7-11 } 5x
16.78	38	14.66	Conv8-12
0.07	26	1.69	DwConv12
8.39	34	9.34	Conv13
0.15	27	2.83	DwConv13
16.78	34	18.37	Conv14
0.00	16	1.20	Pool
1.02	20	15.64	FC

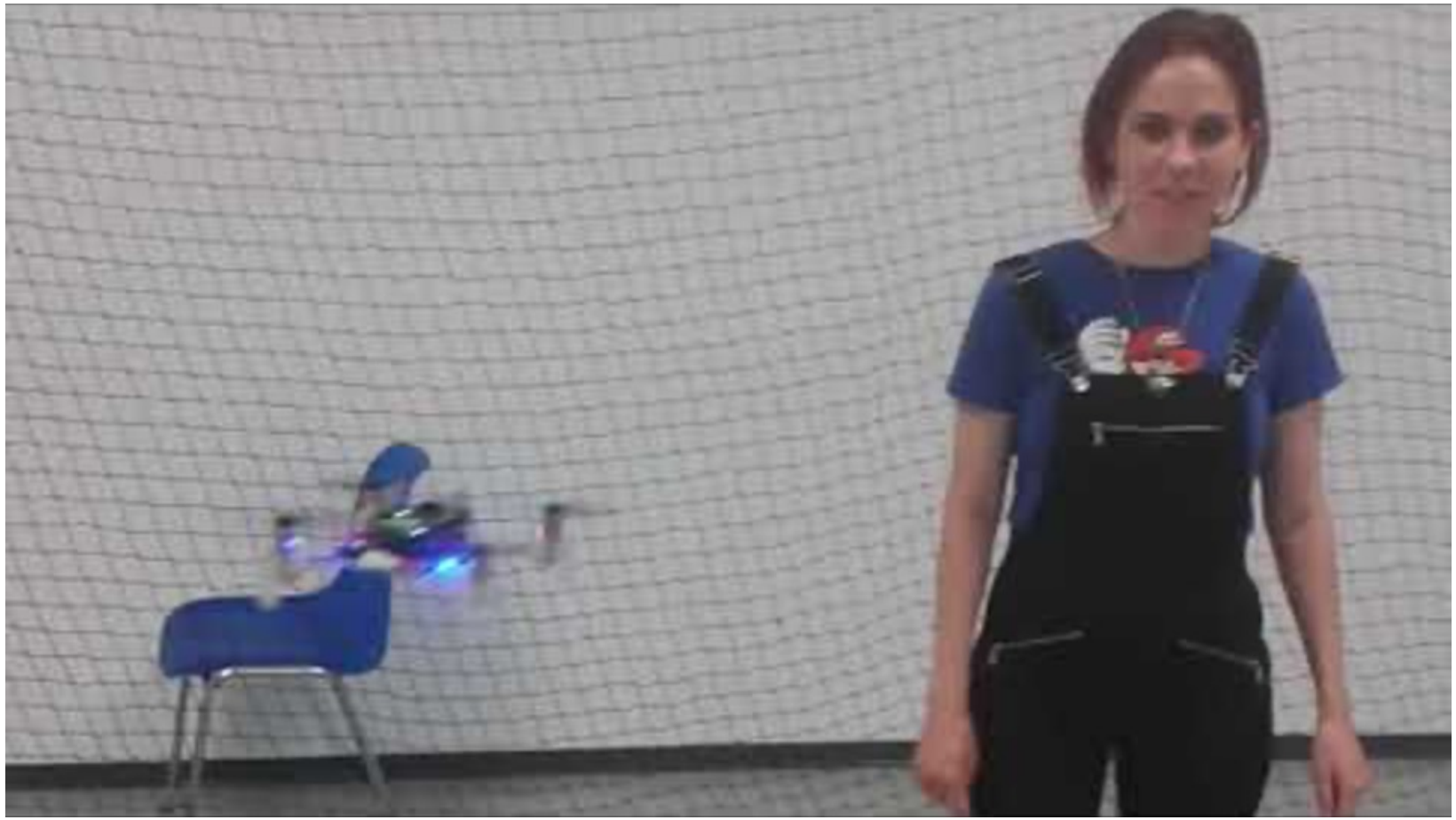


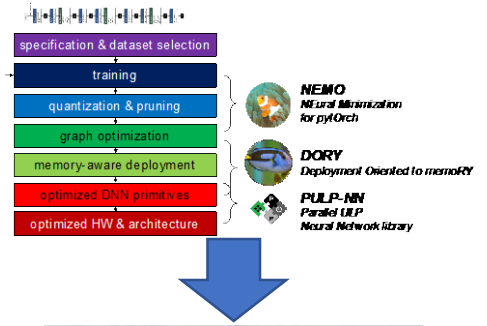
GREENWAVES TECHNOLOGIES

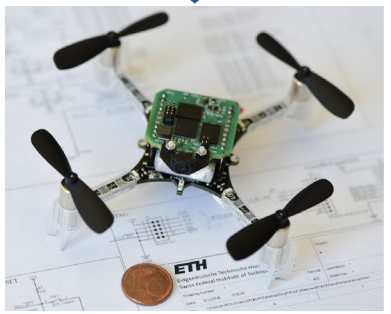
1.0-MobileNet-128
(59% top-1 accuracy on ImageNet)

100 MHz, **4 fps**, **12.5 mJ/frame**
260 MHz, **10 fps**, **25 mJ/frame**

An example of our flow in use!







https://www.youtube.com/watch?v=xBd2nAFNuWY

Credits: Nicky Zimmermann[1], Hanna Müller[2], Jerome Guzzi[1], Alessandro Giusti[1], Daniele Palossi[1,2]
[1]IDSIA Lugano, [2]ETH Zürich

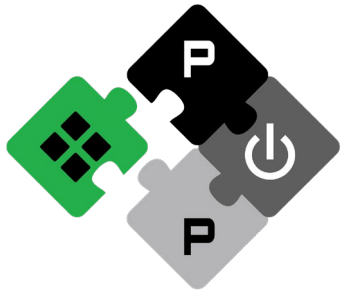
Where can I get it?

Our deployment flow is fully open-source – as is our full hardware platform.

PULP-NN library: https://github.com/**pulp-platform/pulp-nn**

DORY deployment tool: https://github.com/**pulp-platform/dory**

NEMO network minimization tool: https://github.com/**pulp-platform/nemo**



RISC-V® Summit

December 8-10 | Virtual Event

Thanks for your attention!

Credits: Angelo Garofalo, Alessio Burrello, Nazareno Bruschi, Manuele Rusci, Giuseppe Tagliavini, Davide Rossi, Luca Benini & the PULP team

Brought to you by informatech

riscvsummit.com #RISCVSUMMIT