EPFL

EMBEDDED SYSTEMS LABORATORY

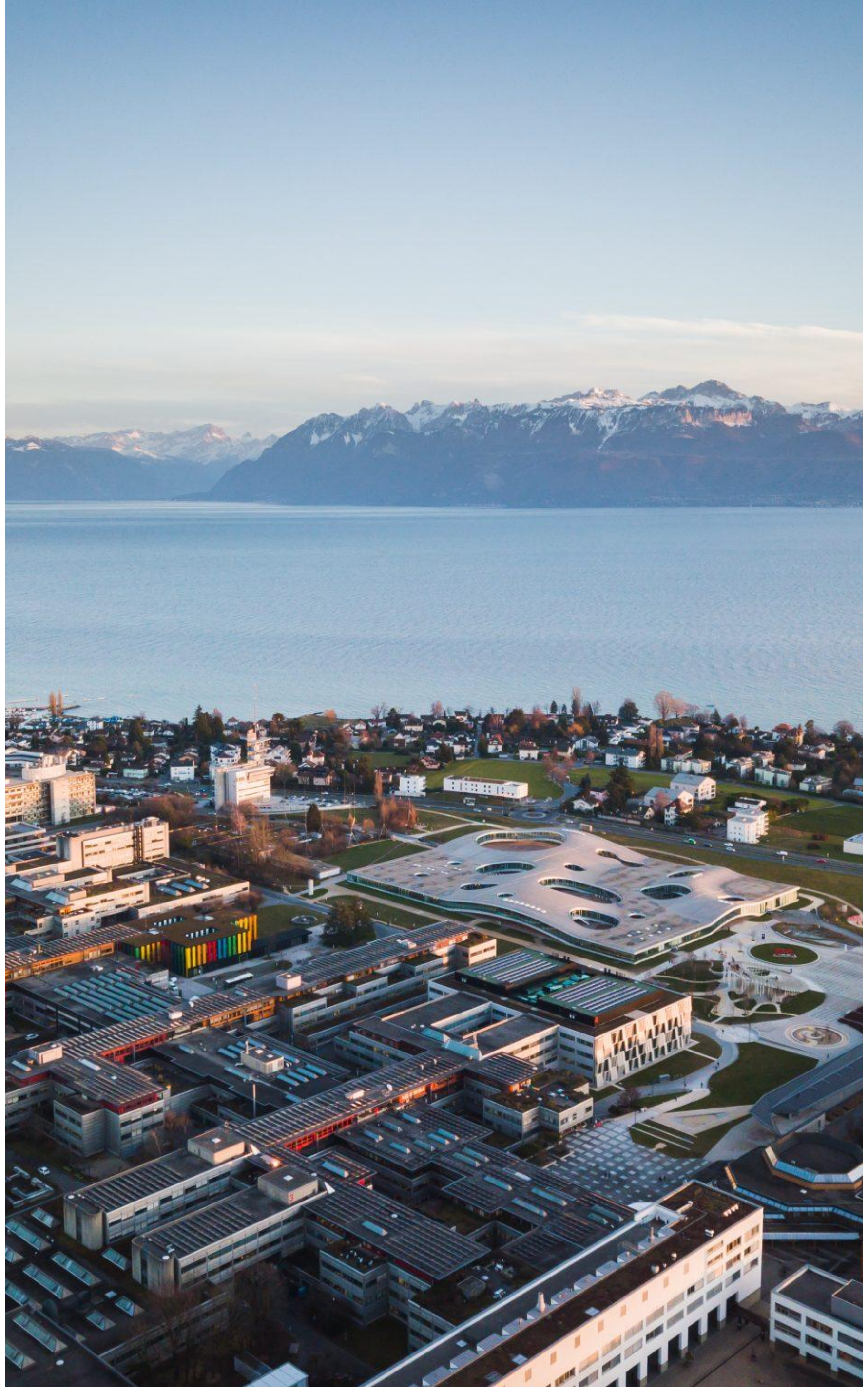
Performance evaluation of acceleration of convolutional layers on OpenEdgeCGRA

Nicolo Carpentieri[1], Juan Sapriza[2], Davide Schiavone[2], Daniele Jahier Pagliari[1], David Atienza[2], Maurizio Martina[1], Alessio Burrello[1]

[1] Politecnico di Torino, Italy
[2] EPFL, Switzerland

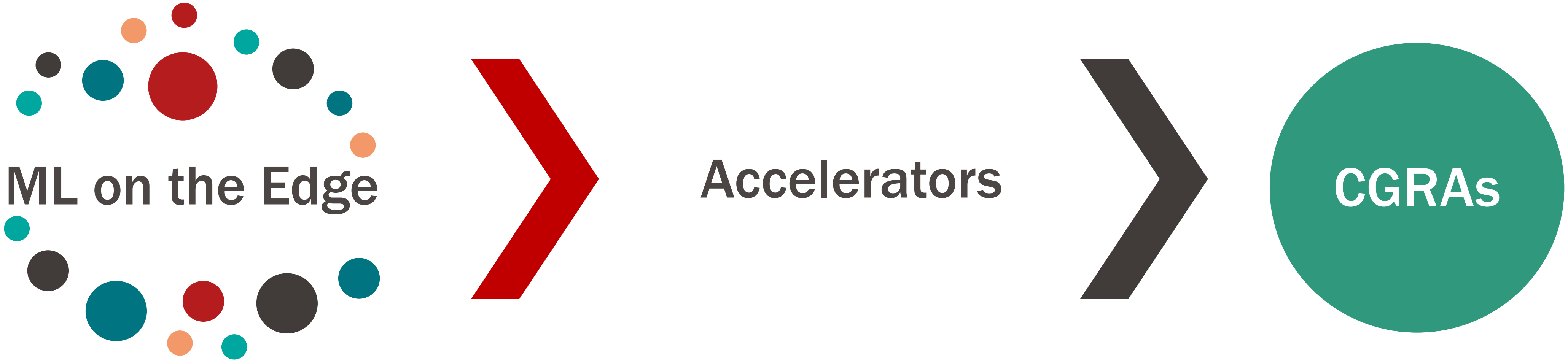
Outline

- Motivation
- Hardware Platform: **HƐƐPsilon**
  - X-HEEP
  - OpenEdgeCGRA
- Methods
  - Input channel parallelism
  - Output channel parallelism
  - Weight parallelism
- Results
- Discussion



Motivation

Motivation



ML on the Edge

Accelerators

CGRAs

- Growing Demand for CNNs in edge devices domain
- Needed for **low-power, high-performance**, and **highly reconfigurable** solutions

- **ASIC** offer **zero reconfigurability**
- **GPUs** consume **significant power** and occupy **large area**
- **FPGA** has **high latency** for **reconfiguration time**

- **Programmable hardware**
- **Low latency** for **reconfiguration time**

Motivation

How do CNNs perform on **time-multiplexed, small, ultra-low-power CGRAs?**

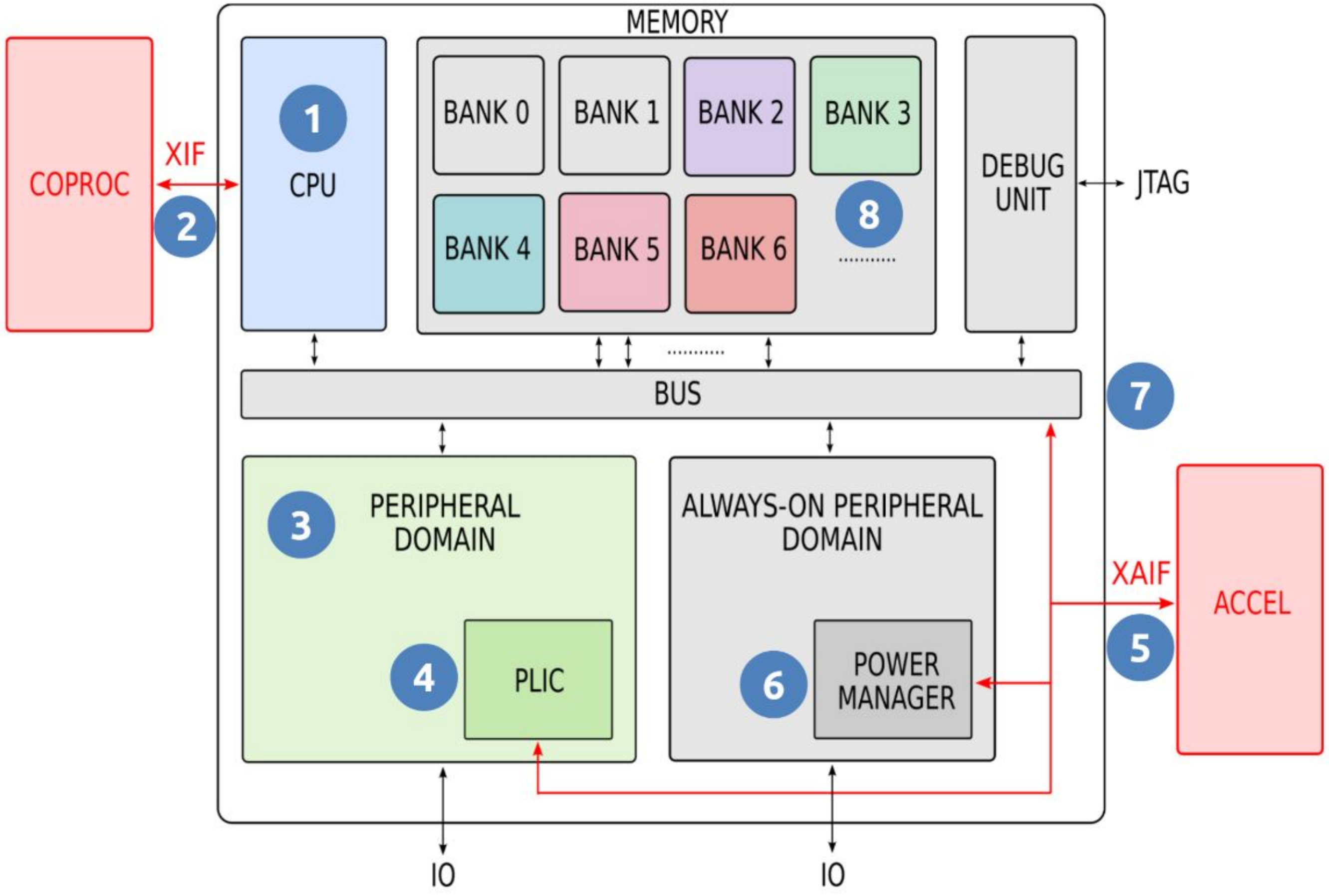
→ Studying different mapping strategies on an Open-Source CGRA

HƐƐPsilon: X-HEEP + OpenEdgeCGRA

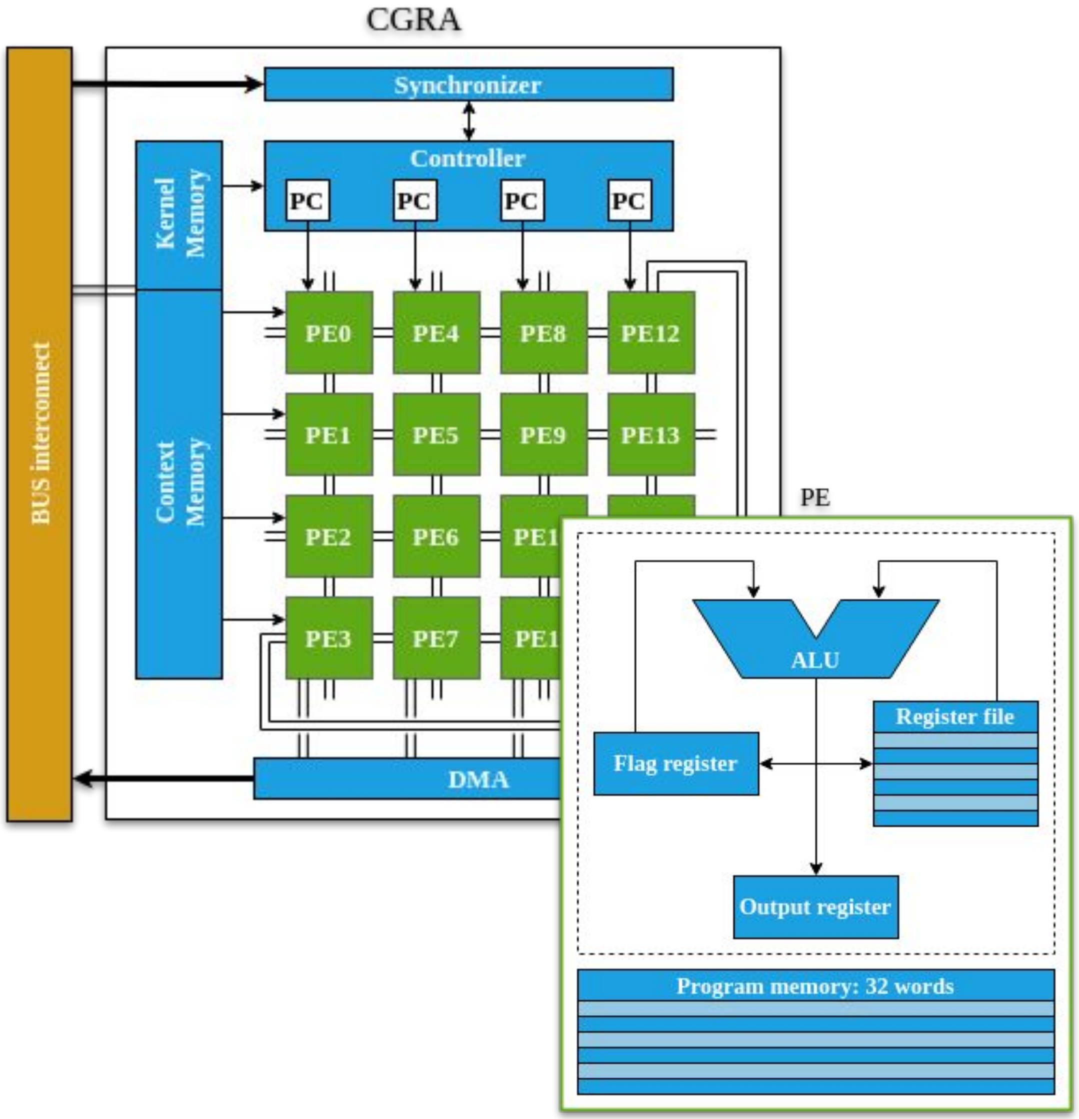
X-HEEP: Ultra-Low-Power Host Platform

Configurability

1. RISC-V core
2. Coprocessor interface
3. Peripherals
4. Interrupt controller
5. Accelerator interface
6. Power manager
7. Bus topology
8. Number of banks



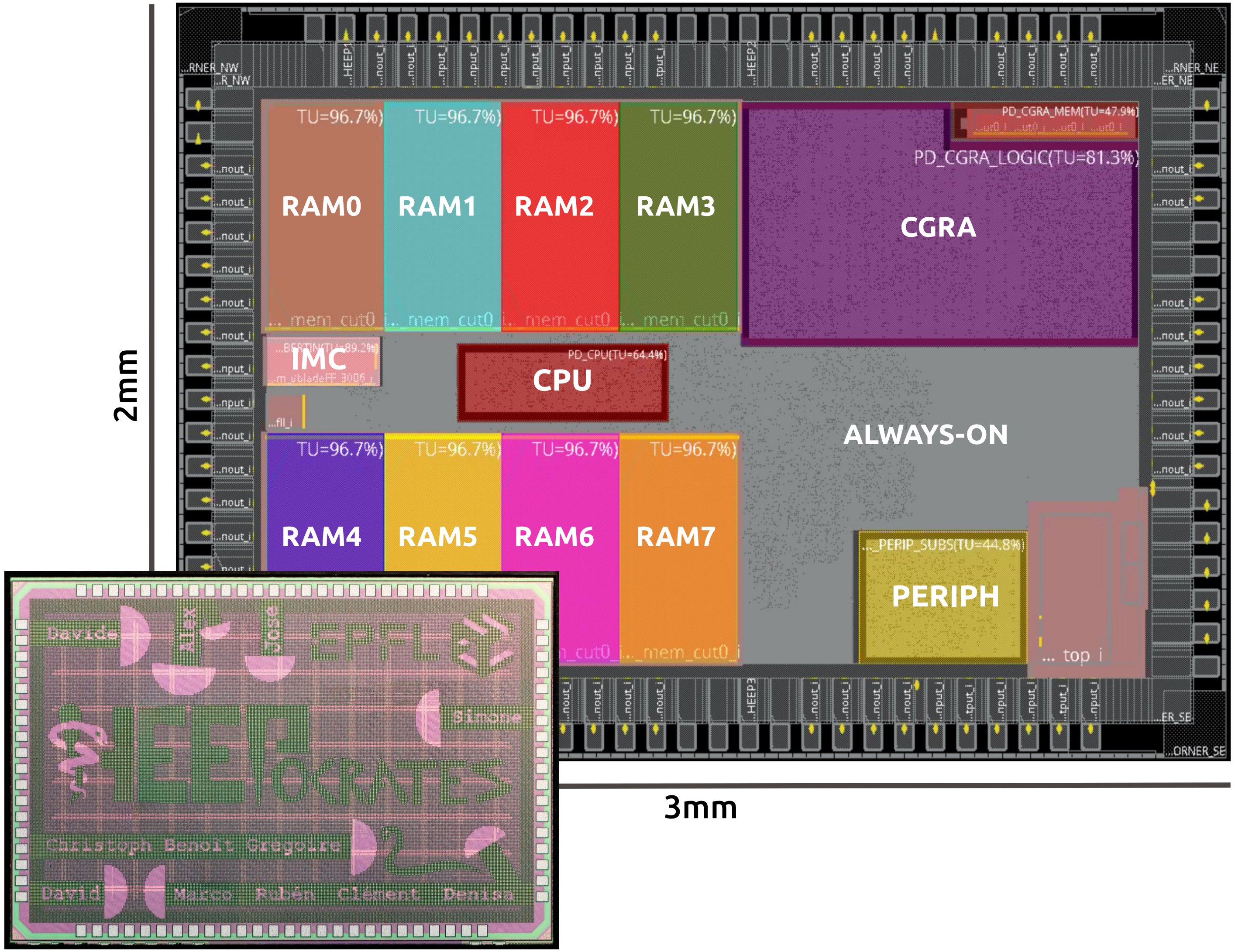
OpenEdgeCGRA



- **4x4** grid of Processing Elements (PEs) interconnected in a torus arrangement.
- Each PE includes
  - ✓ An ALU
  - ✓ Register files (4 + 1 × 32 b)
  - ✓ Private instruction memory (32 inst)

  and executes instructions sequentially.
- Each column includes a DMA r/w port
- Supports diverse kernels with arithmetic, logic, shifts, and **conditional operations.**

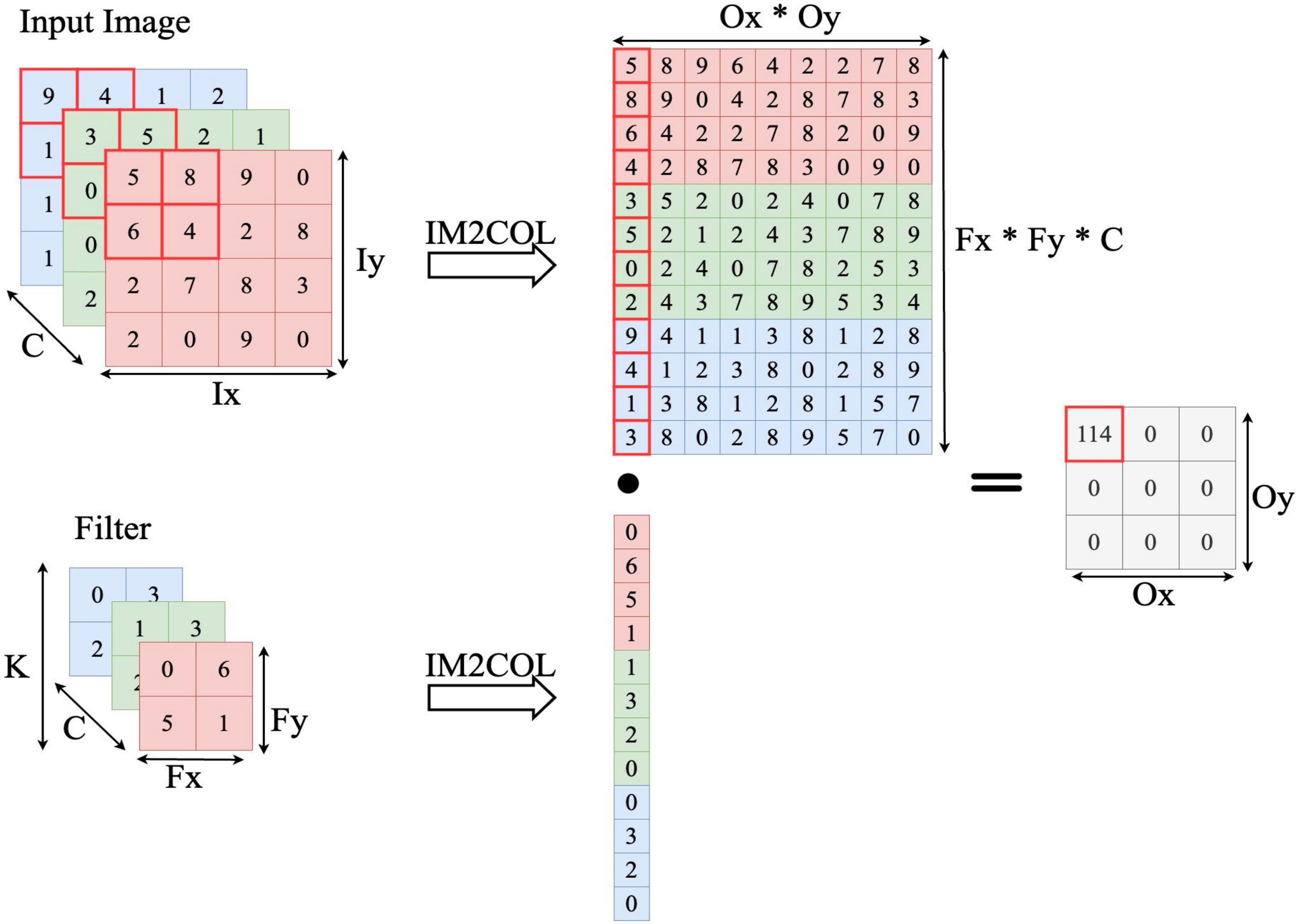
HEEPocrates: First Silicon Prototype



Technology	TSMC 65LP nm CMOS
Area	2x3 mm2
Voltage range	0.8 V - 1.2 V
Frequency range	32 KHz - 170MHz/470 MHz (0.8V/1.2V)
Power range	7.7 mW (170MHz, 0.8V) - 48.1 mW (470 MHz, 1.2 V)

Methods

IM2COL technique



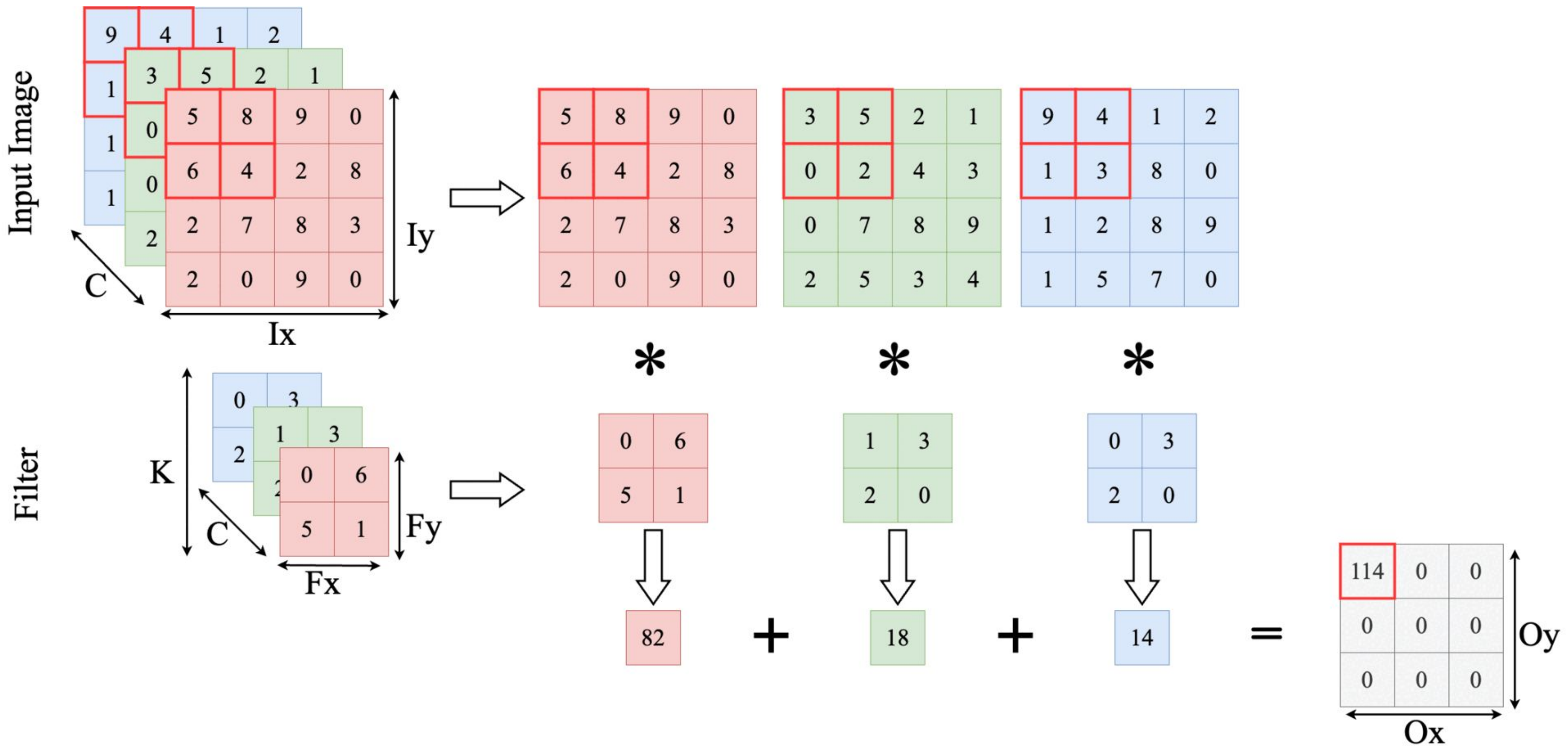
5	8	9	6	4	2	2	7	8
8	9	0	4	2	8	7	8	3
6	4	2	2	7	8	2	0	9
4	2	8	7	8	3	0	9	0
3	5	2	0	2	4	0	7	8
5	2	1	2	4	3	7	8	9
0	2	4	0	7	8	2	5	3
2	4	3	7	8	9	5	3	4
9	4	1	1	3	8	1	2	8
4	1	2	3	8	0	2	8	9
1	3	8	1	2	8	1	5	7
3	8	0	2	8	9	5	7	0

0
6
5
1
1
3
2
0
0
3
2
0

114	0	0
0	0	0
0	0	0

- Used **HWC format** for data organization.
- Ensures sequential memory accesses.
- Requires additional memory

Direct convolution



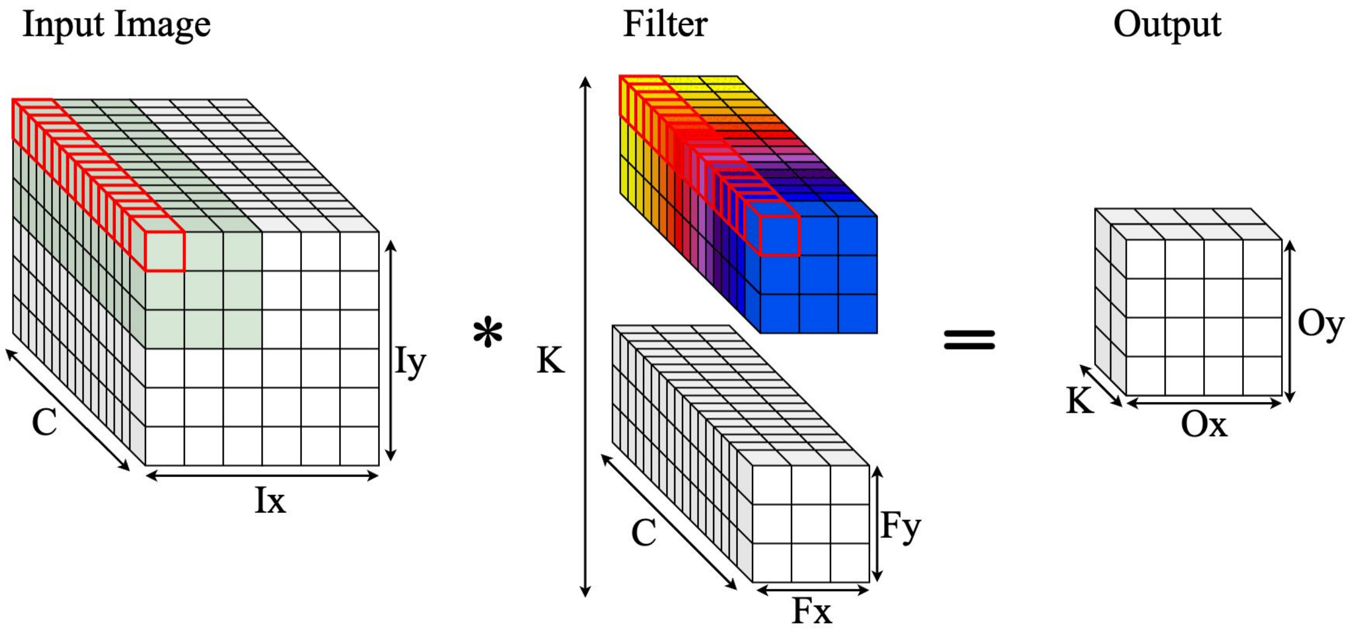
- Used **CHW format** for data organization.
- More overhead when transitioning to a new output row.
- Does not require additional memory

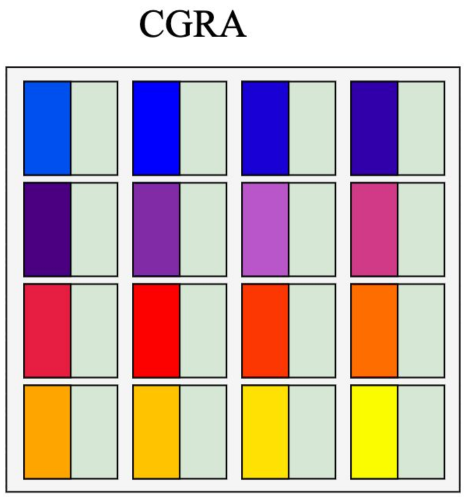
Mappings explored

Three different mappings methods have been developed:

- **Input channel parallelism**
  - No stationarity, assign each PE a different input channel
- **Output channel parallelism**
  - Output stationary, assign each PE a different output channel
- **Weight parallelism**
  - Weight stationary

Input channel parallelism





OPERATION N.2

LOAD INPUT:1

LOAD FILTER:1

MULT:0

ADD:0

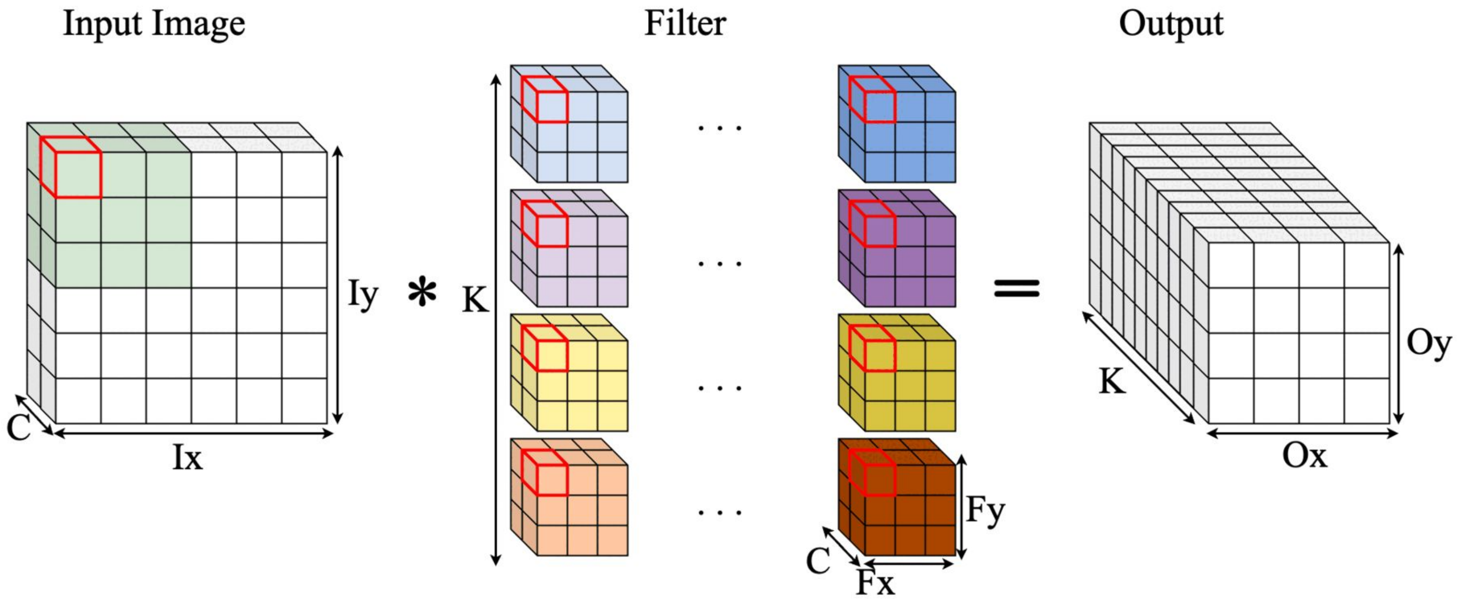
STORE:0

- Used Im2col for sequential access
- Needs 41 operations to store 1 output
- Full utilization: increasing calls (no stationarity).

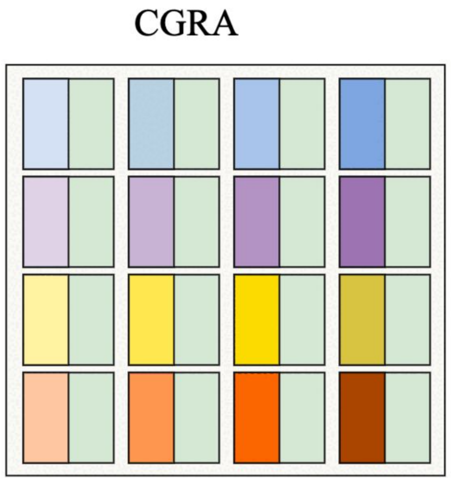
innermost loop over the pixel (with all the input channel)

leveraging spatial connection among PEs to make the final output

Output channel parallelism



- Used Im2col and direct convolution
- Needs 73 operations to store 16 outputs
- Full utilization limits MAC pipelining.



OPERATION N.2

LOAD INPUT:1

LOAD FILTER:1

MULT:0

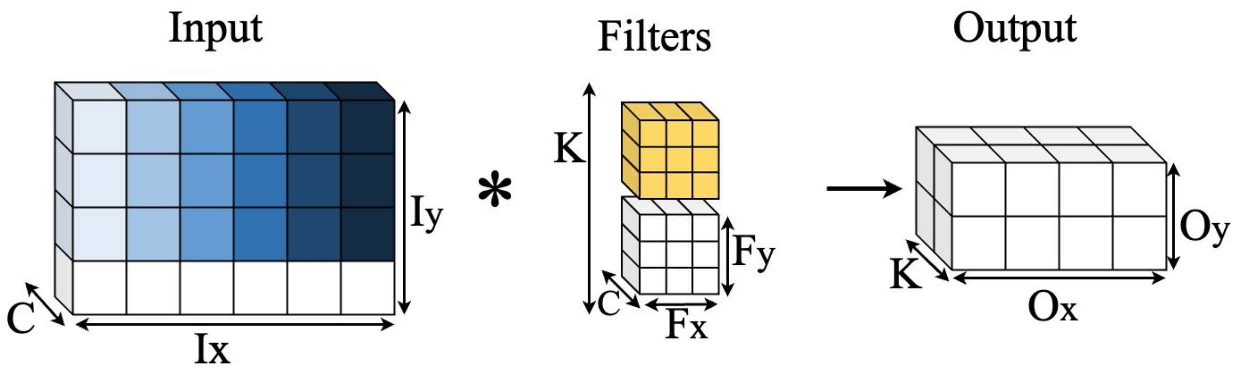
ADD:0

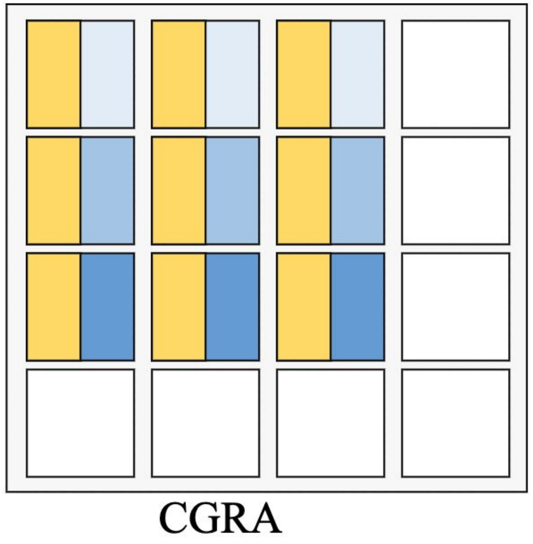
STORE:0

innermost loop iterates over the window

16 independent PEs (one per filter)

Weight parallelism





OPERATION N.1

LOAD:1

MULT:0

ADD:0

STORE:0

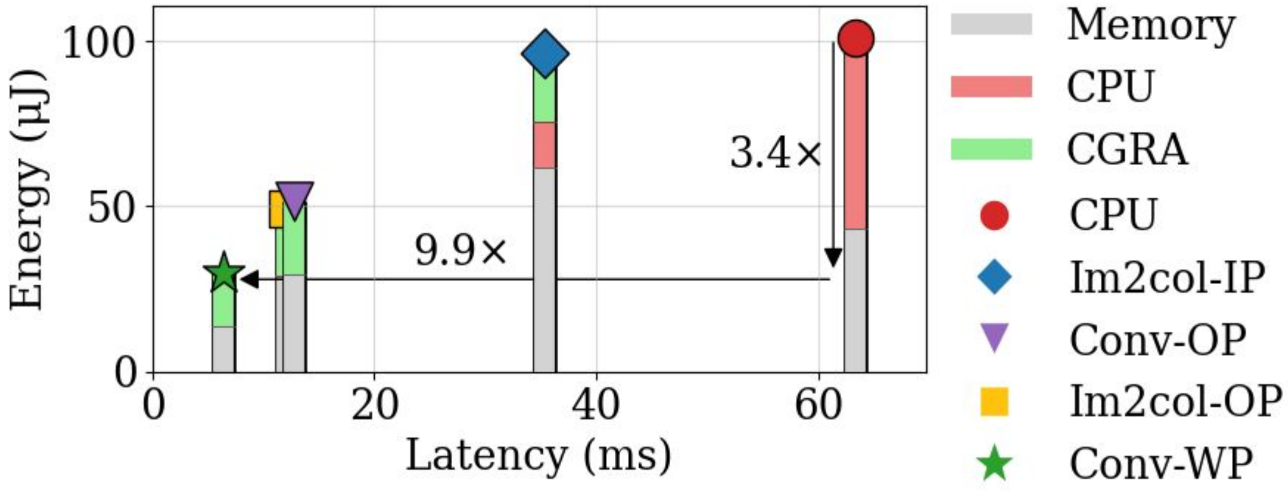
- Assigns **each weight** to a **distinct PE.**
- 9 PEs perform dot products, while others load inputs or sum partial outputs.
- For the first output **7 operations** are needed, then just other **3** for the following ones.

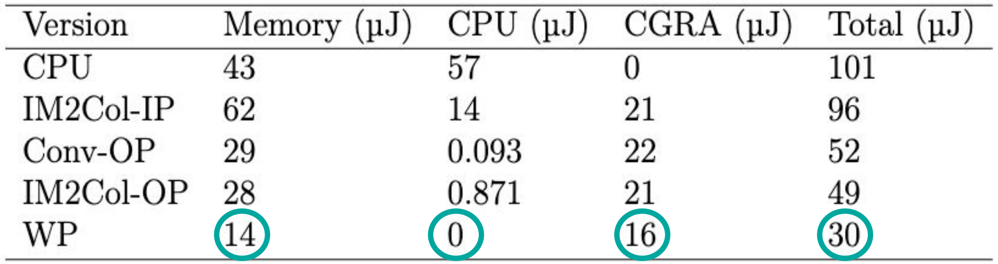
innermost loop slide over the single-channel input window

direct convolution

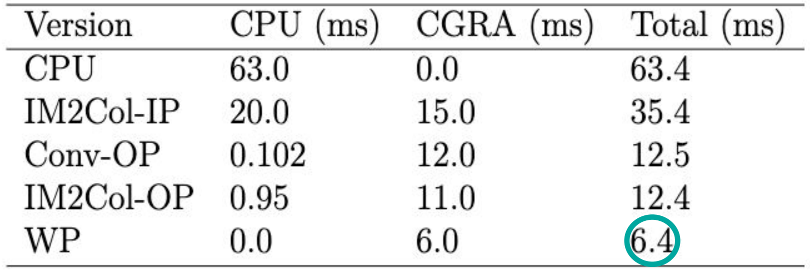
Results

Energy and latency analysis





Version	Memory (µJ)	CPU (µJ)	CGRA (µJ)	Total (µJ)
CPU	43	57	0	101
IM2Col-IP	62	14	21	96
Conv-OP	29	0.093	22	52
IM2Col-OP	28	0.871	21	49
WP	14	0	16	30



Version	CPU (ms)	CGRA (ms)	Total (ms)
CPU	63.0	0.0	63.4
IM2Col-IP	20.0	15.0	35.4
Conv-OP	0.102	12.0	12.5
IM2Col-OP	0.95	11.0	12.4
WP	0.0	6.0	6.4

- All strategies show similar CGRA energy, while the memory use is the most discriminative factor.

- **Frequent Im2col increases latency.**

- **WP: best latency and memory energy**
    - Larger input size allows higher reuse of the loaded weights

Conclusion

- **Weight Parallelism (WP)**
  - ✓ Best latency
  - ✓ Best energy
  - ✓ Most robust
- **CGRA improvements guidelines:**
  - ❑ **ISA extension:** MAC instruction, explicit load/store increment, HW loops
  - ❑ **Data reuse:** Increase number of registers per PE
  - ❑ **Parallelism:** Using interleaved memory

Thank you!

Davide Schiavone

EPFL - Embedded Systems Laboratory
davide.schiavone@epfl.ch