TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Binyi Wu
Faculty of Electrical Engineering and Information Technology
Chair of Highly-Parallel VLSI Systems and Neuro-Microelectronics

Efficient Mapping of Convolutional Neural Networks on SpiNNaker2 prototype

Dresden, 29.05.2019

Content

- Motivation
- SpiNNaker2 and Simulator: SpiNNaker2Py
- Mapping Strategy
- Validation and Simulation
- Conclusion

Content

- Motivation
- SpiNNaker2 and Simulator: SpiNNaker2Py
- Mapping Strategy
- Validation and Simulation
- Conclusion

Motivation

- SpiNNaker2:
  1. 144 Processing elements (PEs)
  2. PE: ML-accelerator with 64 MACs but limited SRAM (128 KB)
- CNN:
  1. Every layer of state-of-art model is very large
     (VGG-CONV_2 → input: 3 MB, weight: 36 KB, output: 6 MB)

⇒

- dedicated mapping strategies are need.
  1. Layers in CNN → primitive operations supported by SpiNNaker2
  2. How to chain different operations?
  3. How to split each operation?
  4. How to distribute into SpiNNaker2?

- SpiNNaker2 is still under development

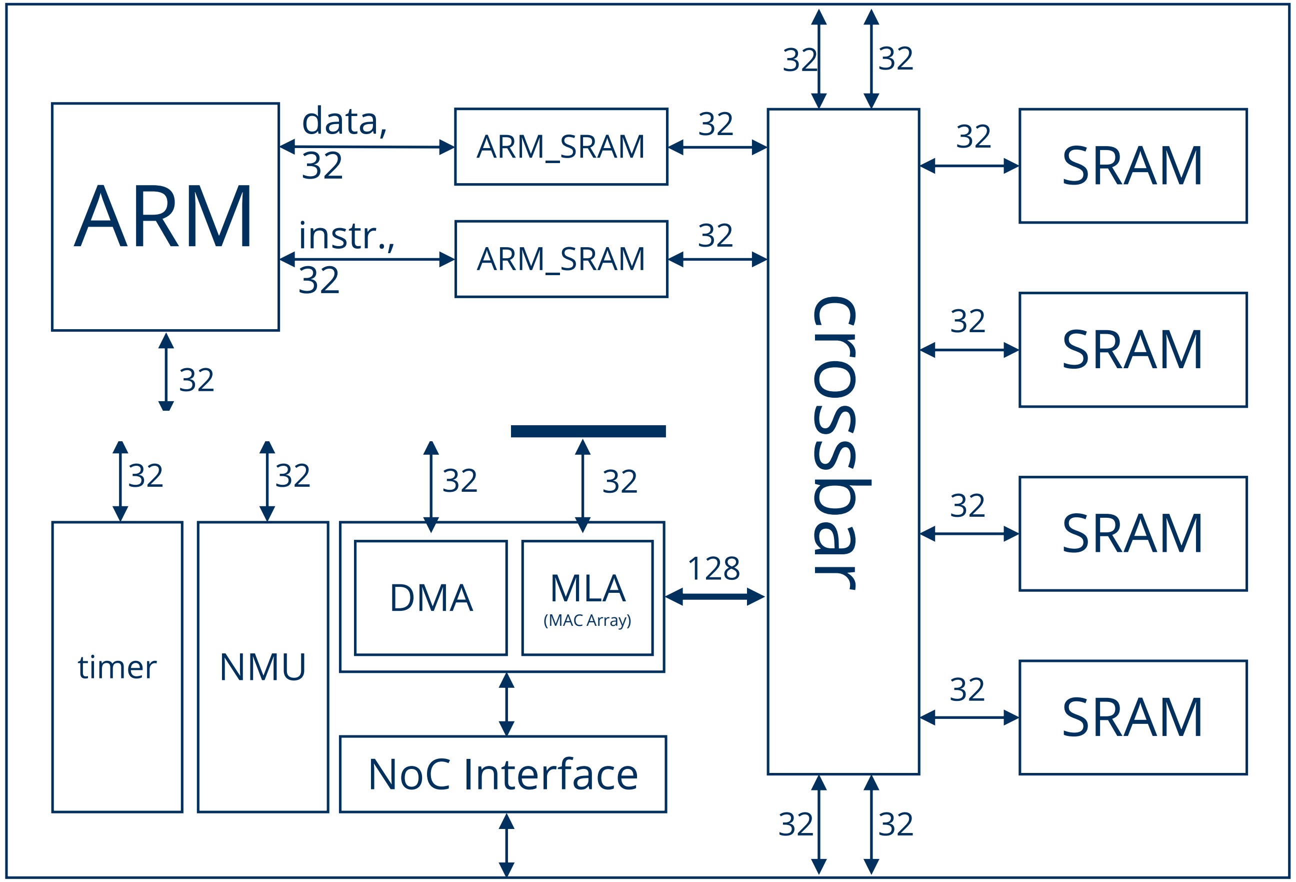
⇒

- SpiNNaker2 simulator (SpiNNaker2Py)

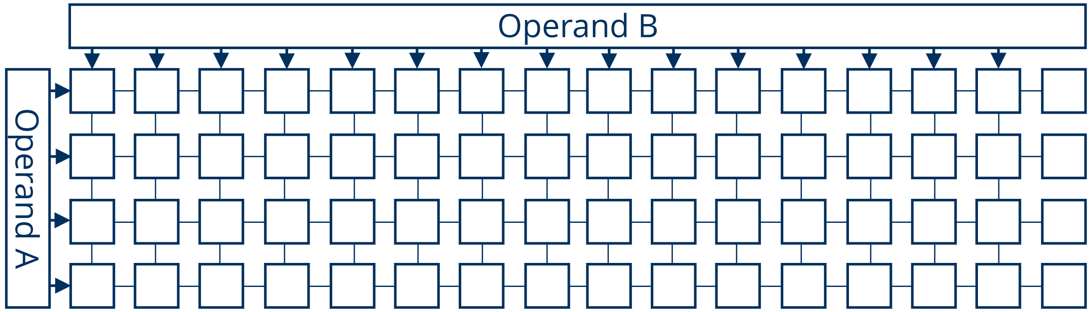
Content

- Motivation
- SpiNNaker2 and Simulator: SpiNNaker2Py
- Mapping Strategy
- Validation and Simulation
- Conclusion

SpiNNaker2: PE



PE: Processing element

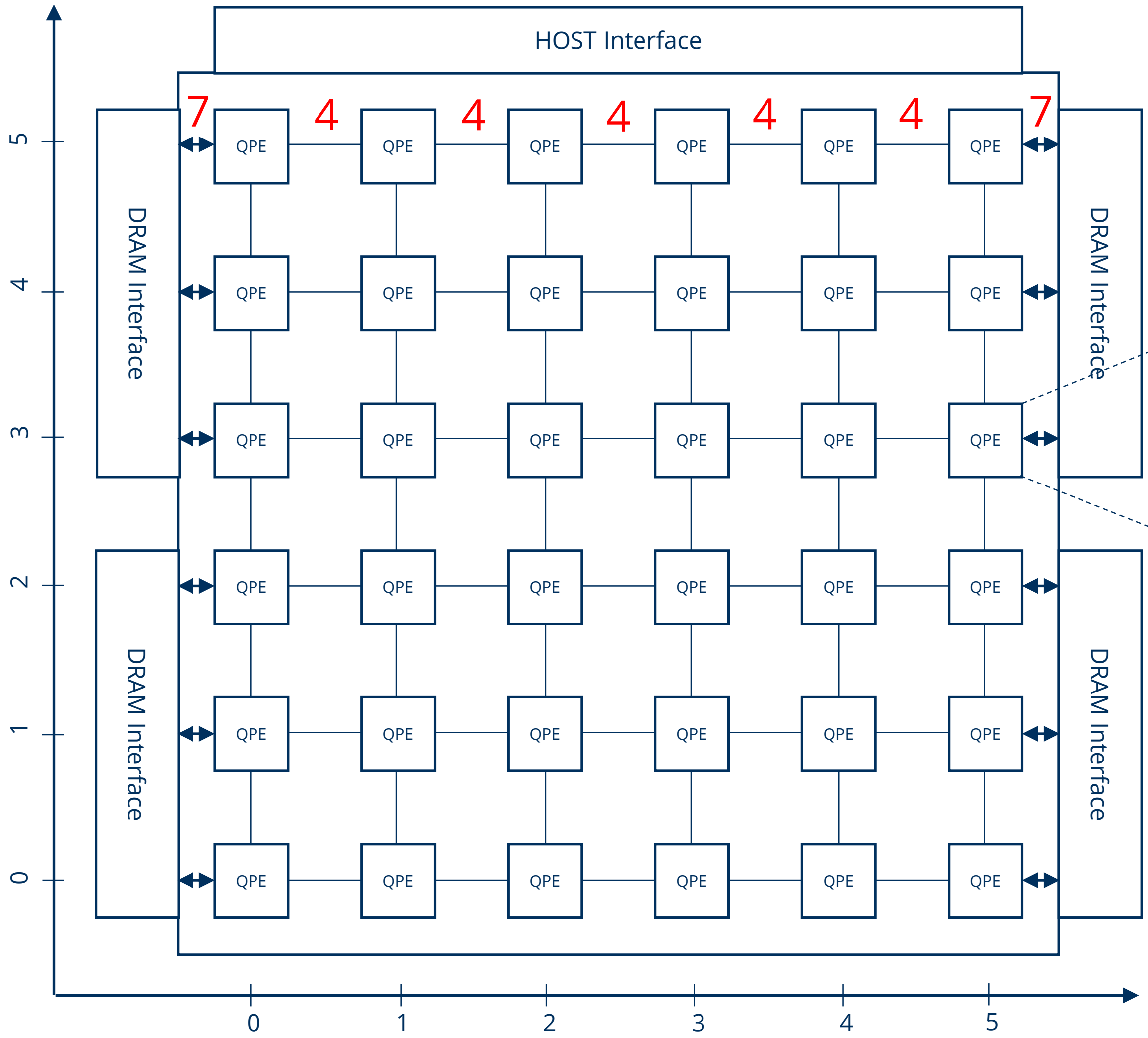


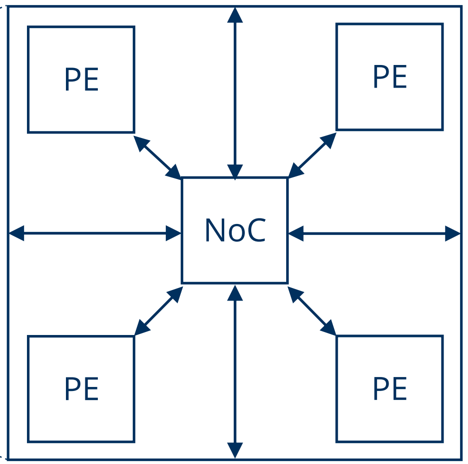
Operand A: From local PE SRAM or neighbor PE SRAM through NoC

Operand B: From local PE SRAM

[2] TU Dresden. SpiNNaker2 Wiki: SpiNNaker2 Universal Spiking Neural Network Architecture

SpiNNaker2 and QPE



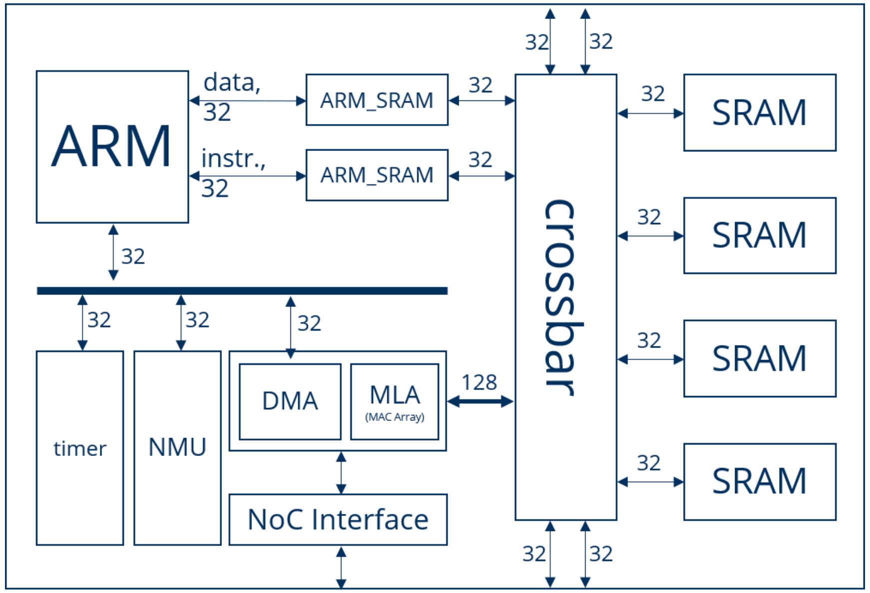


Quad-PE

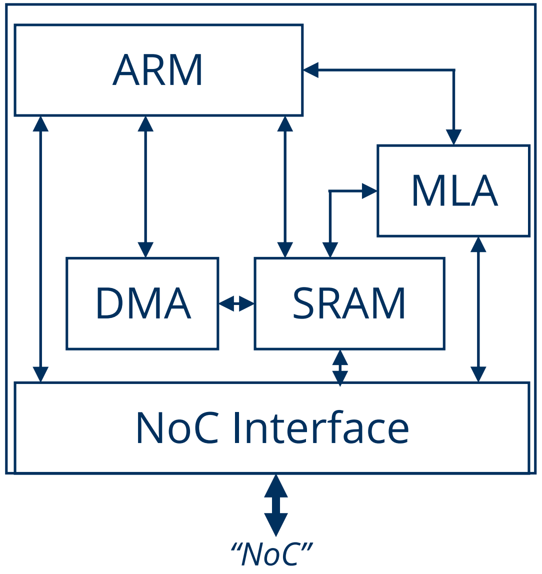
- 36 QPEs (144) PEs
- 4 DRAMs

[2] TU Dresden. SpiNNaker2 Wiki: SpiNNaker2 Universal Spiking Neural Network Architecture

SpiNNaker2 Simulator: PE simulator



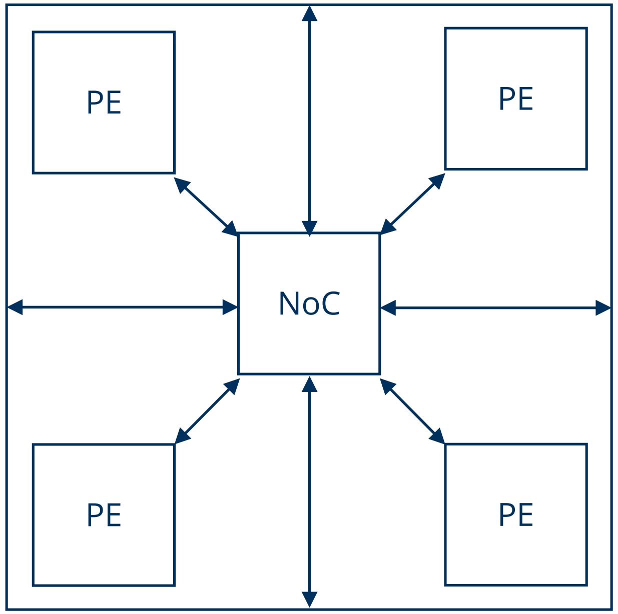
Processing element (PE)



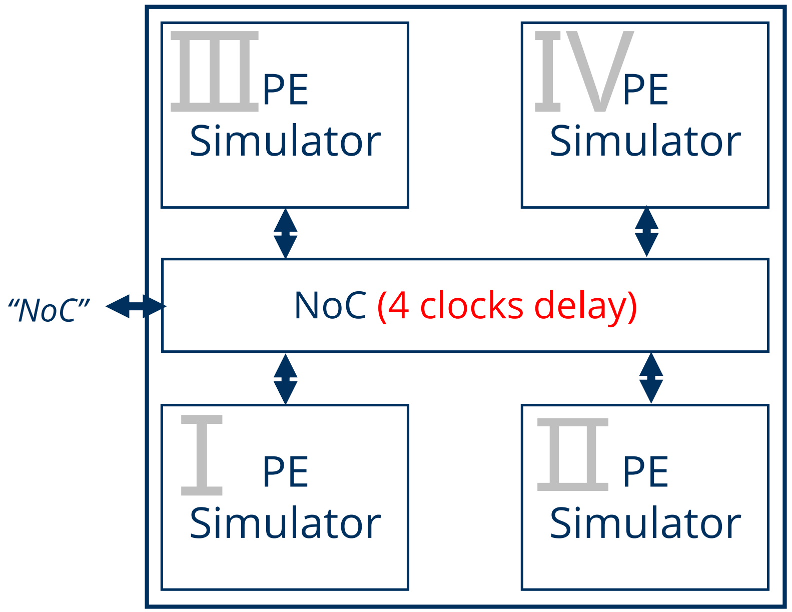
No timer and NMU

Processing element (PE) Simulator

SpiNNaker2 Simulator: QPE simulator



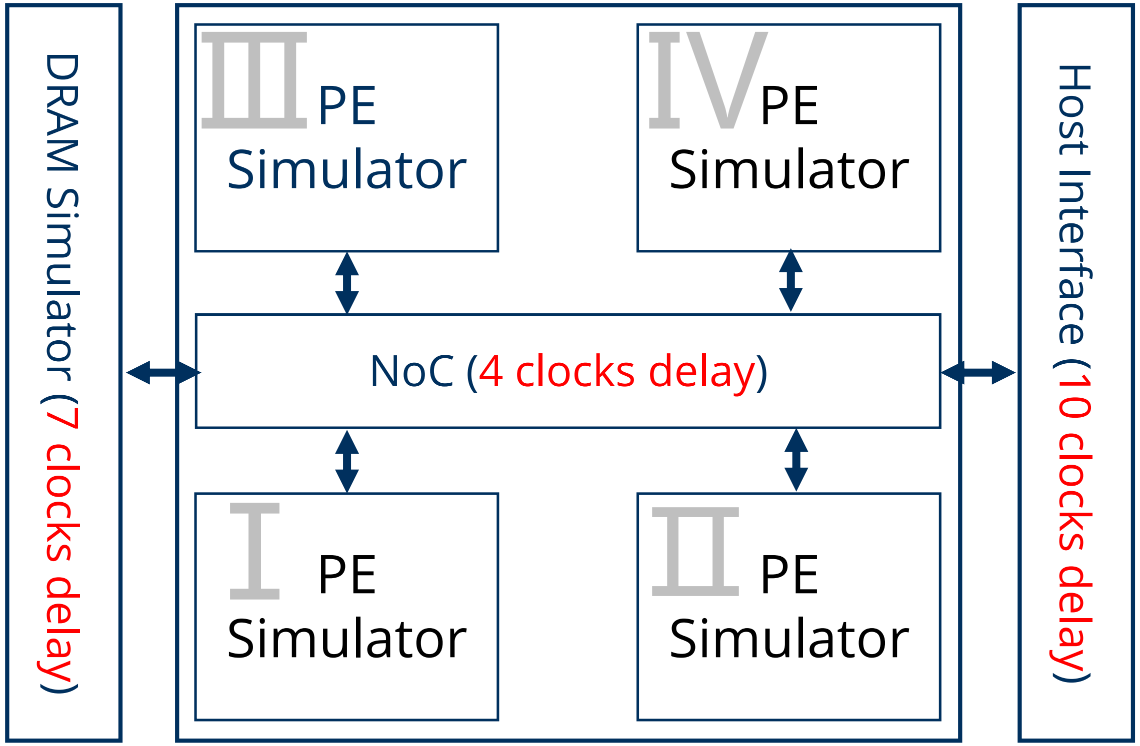
QPE



QPE Simulator

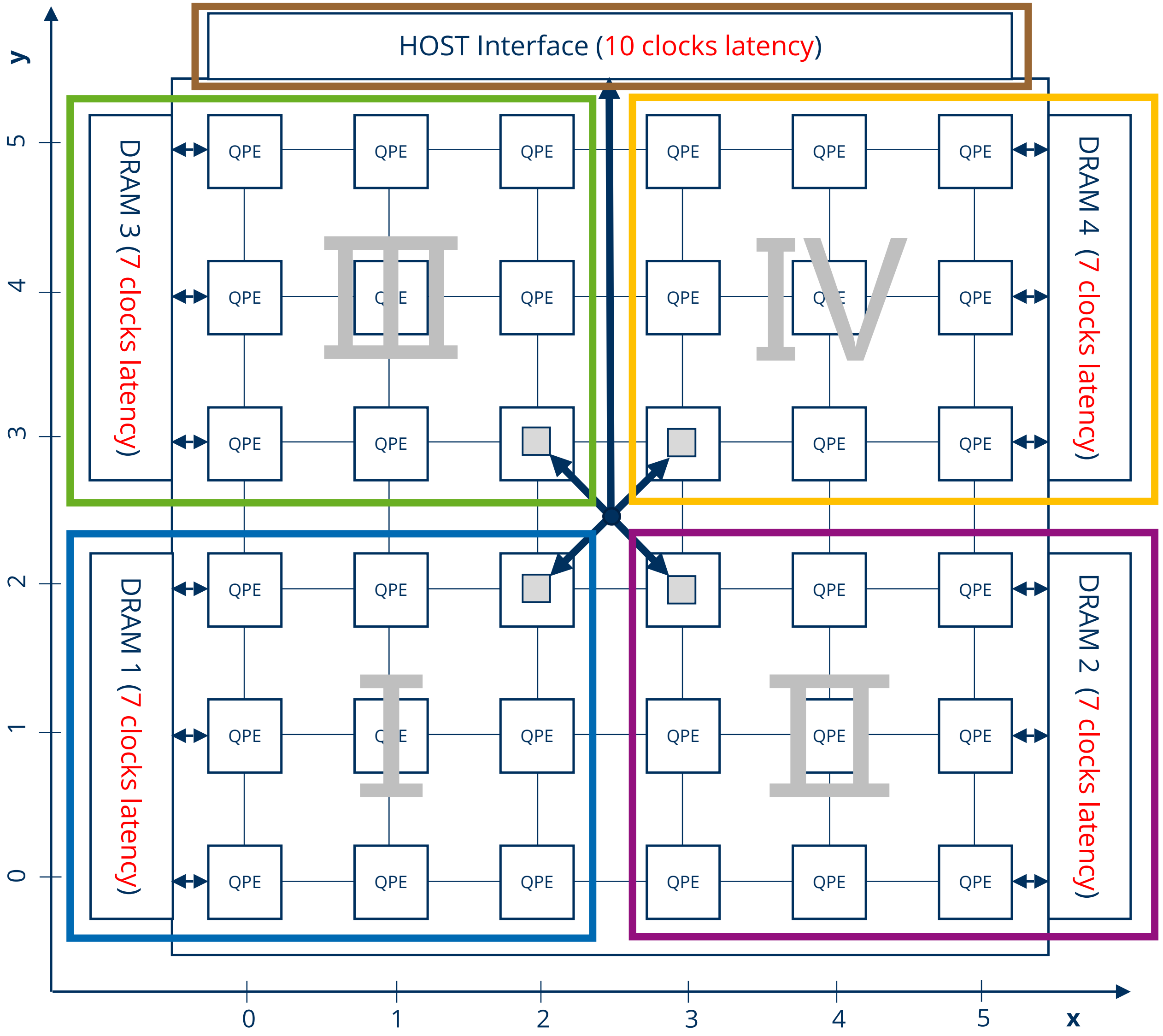
[2] TU Dresden. SpiNNaker2 Wiki: SpiNNaker2 Universal Spiking Neural Network Architecture

SpiNNaker2 Simulator: QPE-DRAM simulator



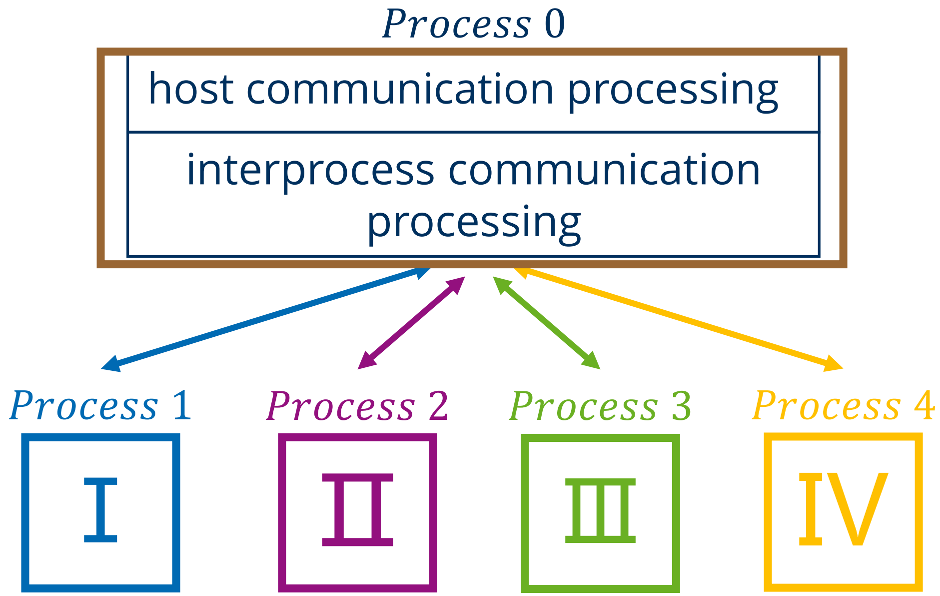
QPE-DRAM simulator

SpiNNaker2 Simulator: SpiNNaker2 simulator



SpiNNaker2 simulator

Accelerating the simulation through **multi-processing**



Content

- Motivation
- SpiNNaker2 and Simulator: SpiNNaker2Py
- Mapping Strategy
- Validation and Simulation
- Conclusion

Mapping Strategy

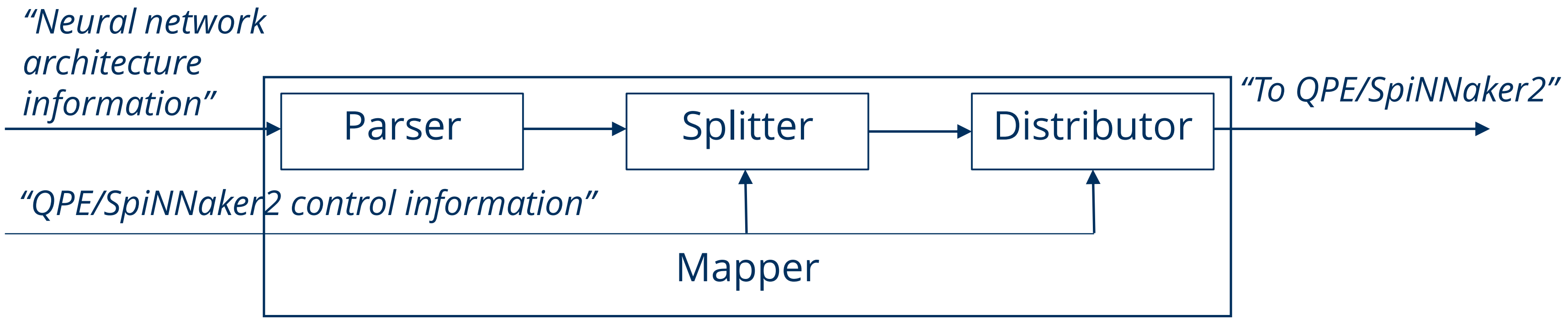
dedicated mapping strategies:

1. Layers in CNNs → primitive operations of SpiNNaker2
2. How to chain different operations?
3. How to split each operation?
4. How to distribute into SpiNNaker2?

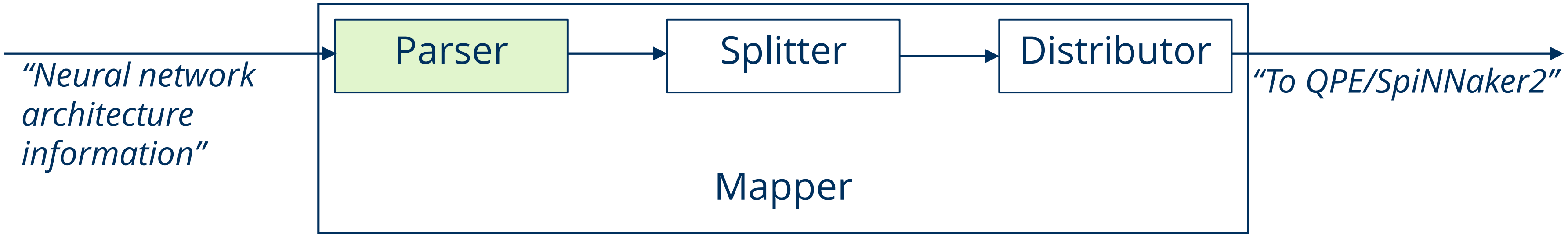
1.–2. → Parser

3. → Splitter

4. → Distributor



Mapping Strategy: Parser



- Parse the neural network
- Layer → Operations e.g.

convolutional layer	*padding operation (ARM)*
	convolution operation (MLA)
	nonlinearity operation (ARM)
	quantization operation (ARM)

- Operator fusion → operation blocks

Mapping Strategy: Parser

Operator fusion → operation block

1. convolution block

$$\textbf{core operation} \leftarrow \begin{bmatrix} padding\ operation\ (ARM) \\ \boldsymbol{convolution\ operation(MLA)} \\ nonlinearity\ operation(ARM) \\ quantization\ operation(ARM) \end{bmatrix} \longrightarrow convolution\ block$$

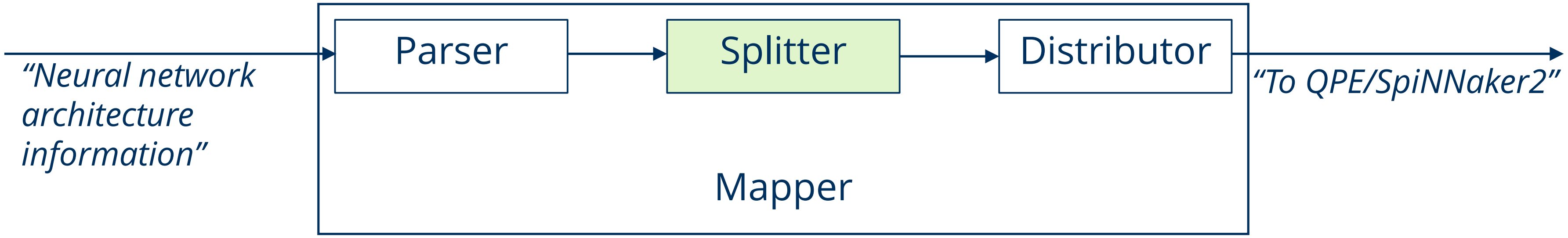
2. pooling block ← *(stride not <u>equal to</u> pooling width/height)*

$$\textbf{core operation} \leftarrow \begin{bmatrix} padding\ operation(ARM) \\ \boldsymbol{pooling\ operation(ARM)} \end{bmatrix} \longrightarrow pooling\ block$$

3. matrix multiplication block

$$\textbf{core operation} \leftarrow \begin{bmatrix} \boldsymbol{matrix\ multiplication\ operation(MLA)} \\ nonlinearity\ operation\ (ARM) \\ quantization\ opoeration\ (ARM) \end{bmatrix} \longrightarrow matrix\ multiplication\ block$$

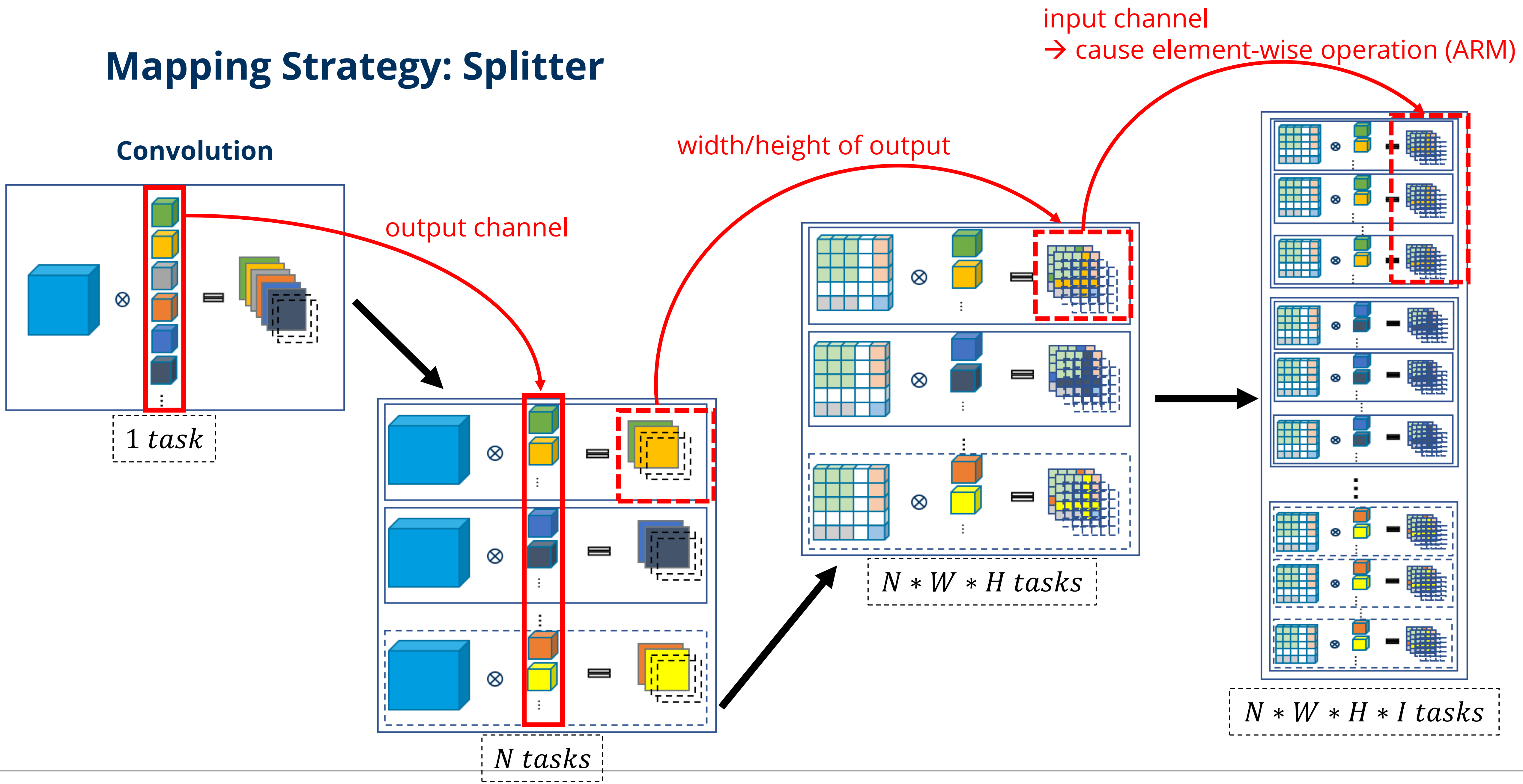
Mapping Strategy: Splitter



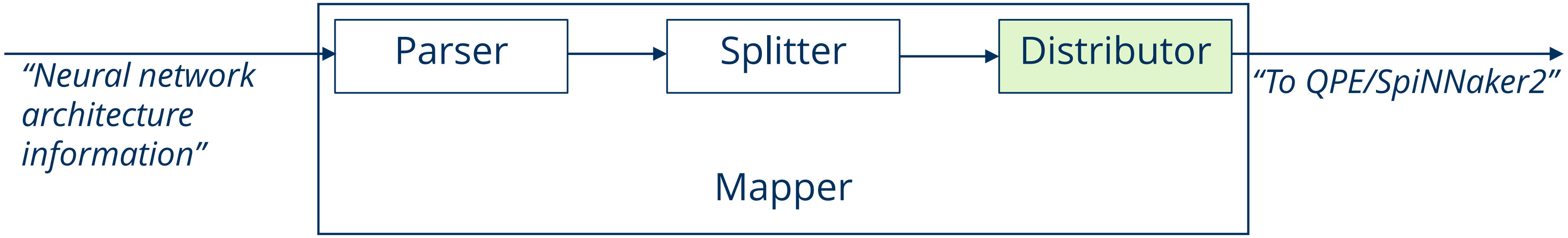
- Split the core operation
  1. Convolution block → convolution operation
  2. Pooling block → pooling operation
  3. Matrix multiplication block → MM operation

- SRAM utilization, MAC utilization, PE utilization, size increasement by splitting, computation balance and acceleration speed are considered during splitting.

Mapping Strategy: Splitter



Mapping Strategy: Distributor



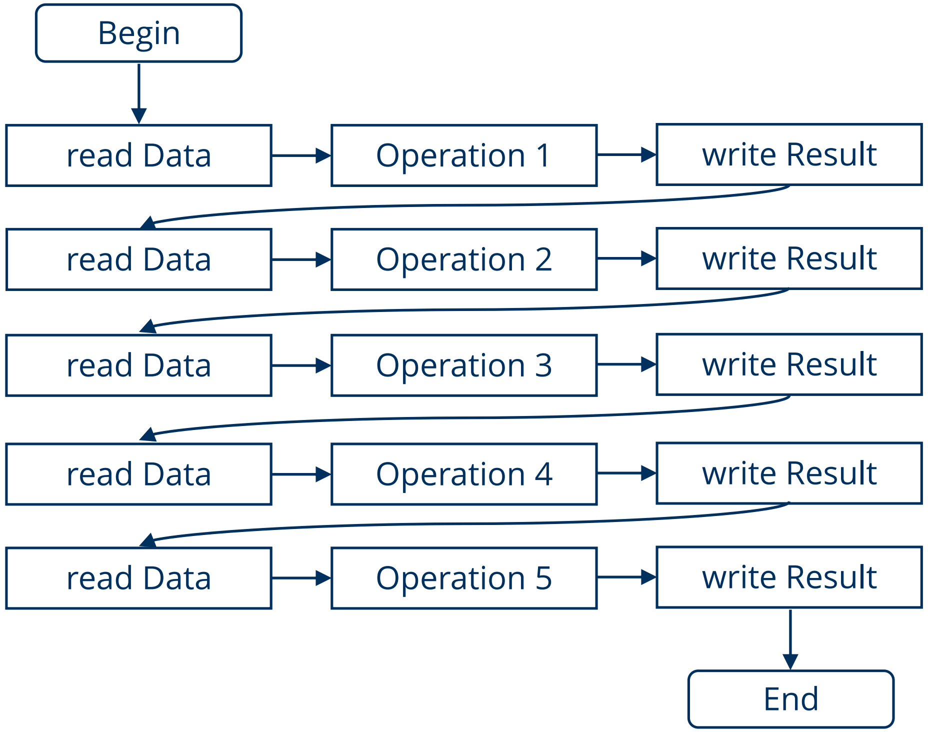
3 distribution algorithms

- Without operator fusion and without data reuse

Mapping Strategy: Distributor

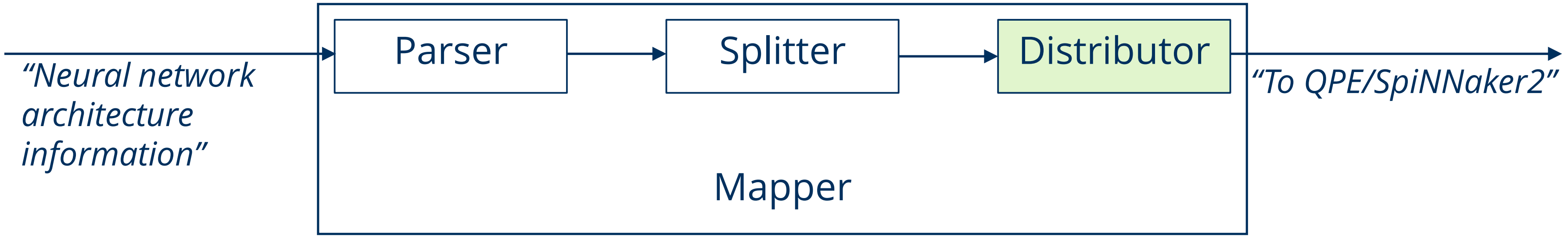
Without operator fusion and without data reuse

- Each PE runs entirely independently from other PEs. Once a PE has completed its work, it writes out the result and immediately get a new task.



Without operator fusion

Mapping Strategy: Distributor



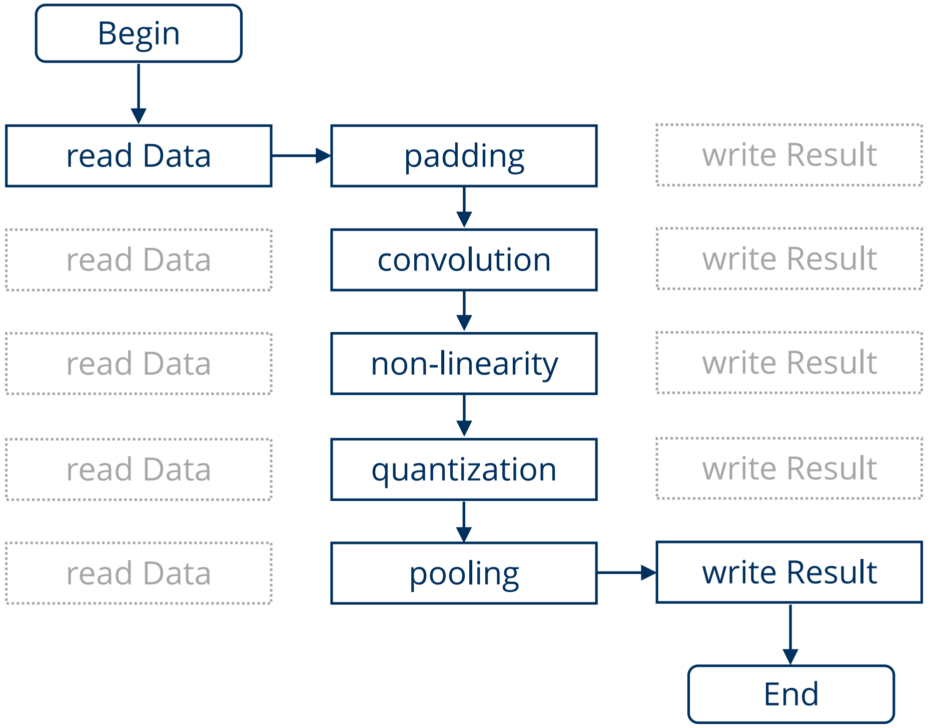
3 distribution algorithms

- Without operator fusion and without data reuse
- With operator fusion and without data reuse

Mapping Strategy: Distributor

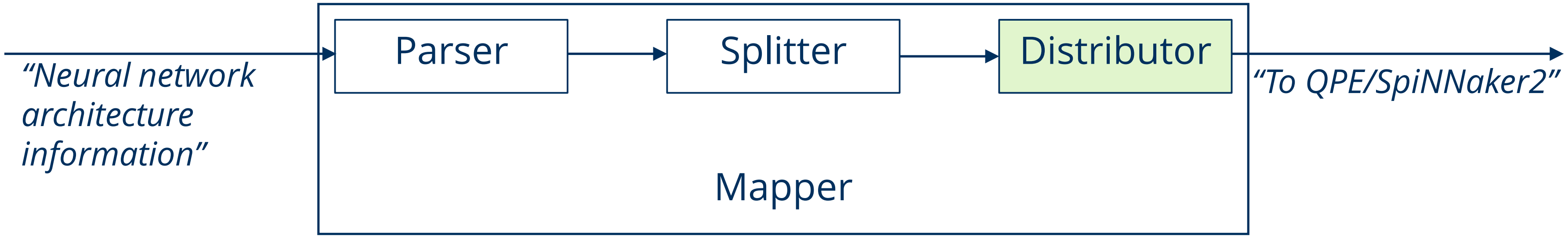
With operator fusion and without data reuse

- Each PE runs entirely independently from other PEs. Once a PE has completed its work, it writes out the result and immediately get a new task.
- Take operation block into account



With operator fusion (convolution block)

Mapping Strategy: Distributor



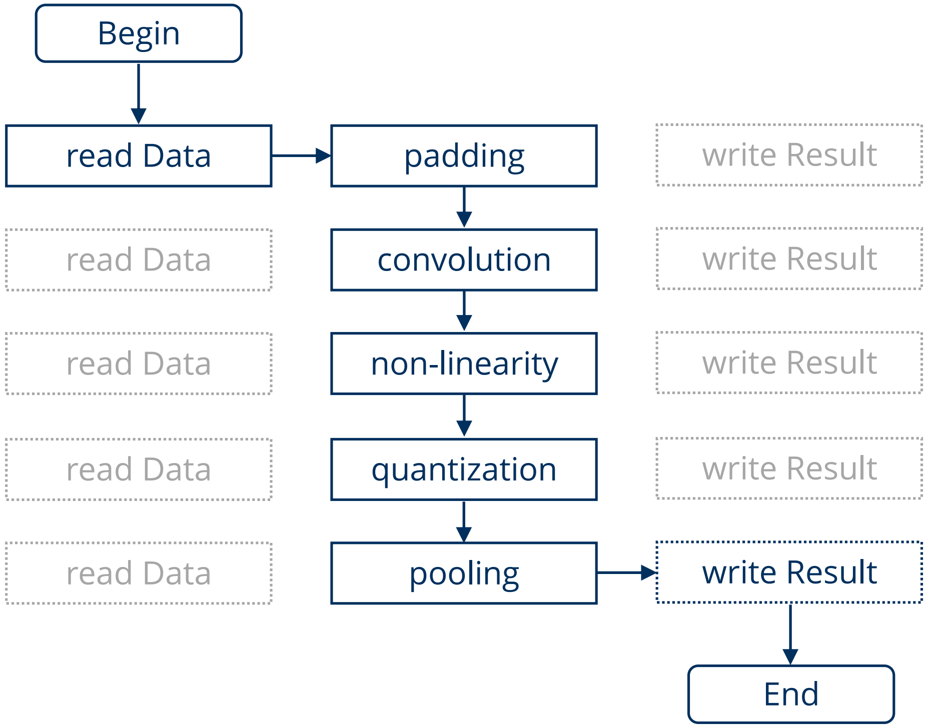
3 distribution algorithms

- Without operator fusion and without data reuse
- With operator fusion and without data reuse
- With operator fusion and with data reuse

Mapping Strategy: Distributor

With operator fusion and with data reuse

- Only convolution operation has data reuse.
- All Pes relate to each other !!
- Different for QPE and SpiNNaker2



With operator fusion (convolution block)

Mapping Strategy: Distributor

Data reuse in QPE

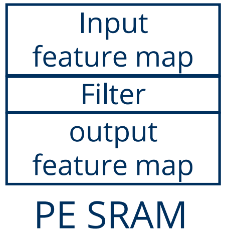
1. Update F_1, F_2, F_3, F_4 to F_5, F_6, F_7, F_8

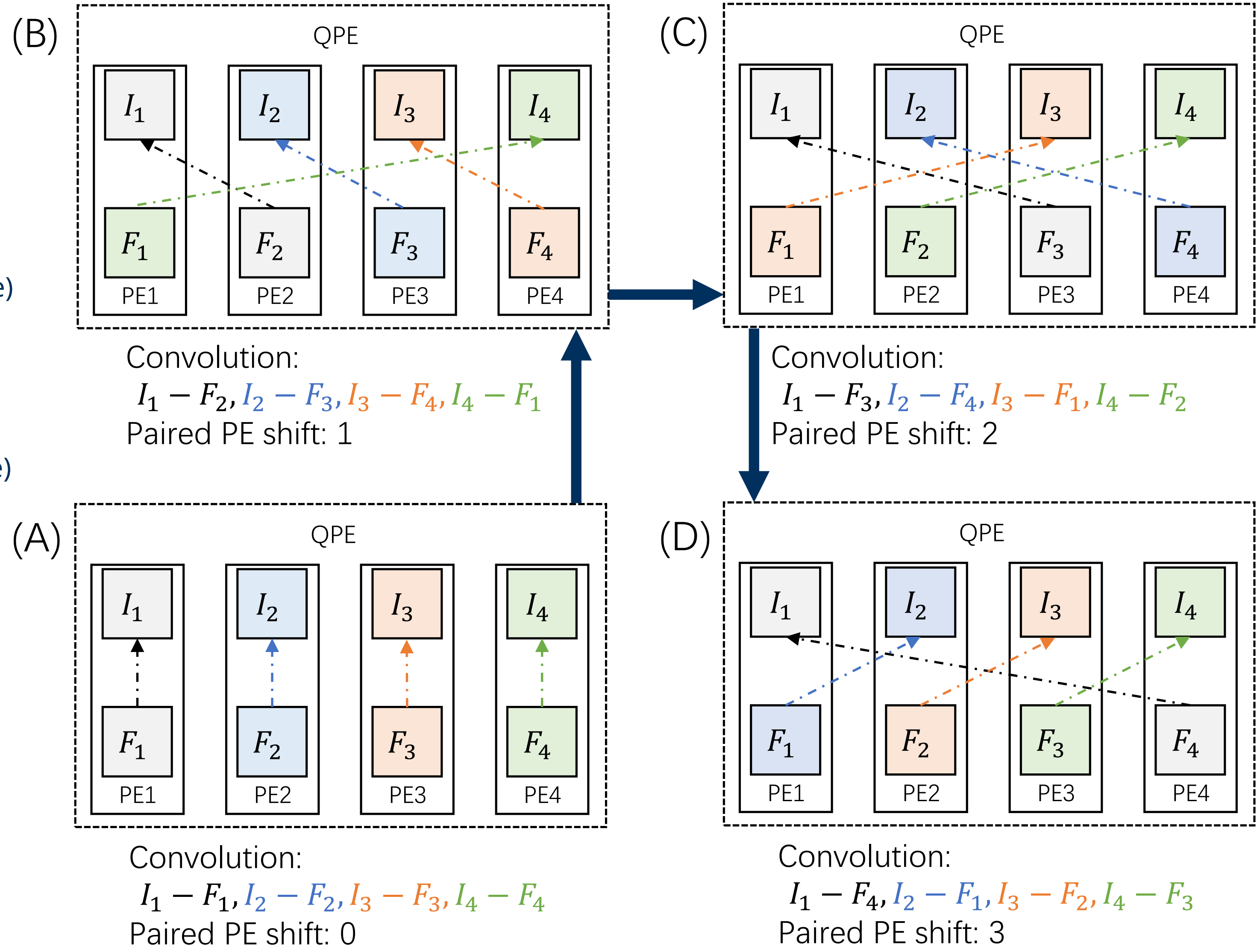
→Feature map reuse (partial filter weight reuse)

2. Update I_1, I_2, I_3, I_4 to I_5, I_6, I_7, I_8

→Filter weight reuse (partial feature map reuse)

Selected based on the data amount.



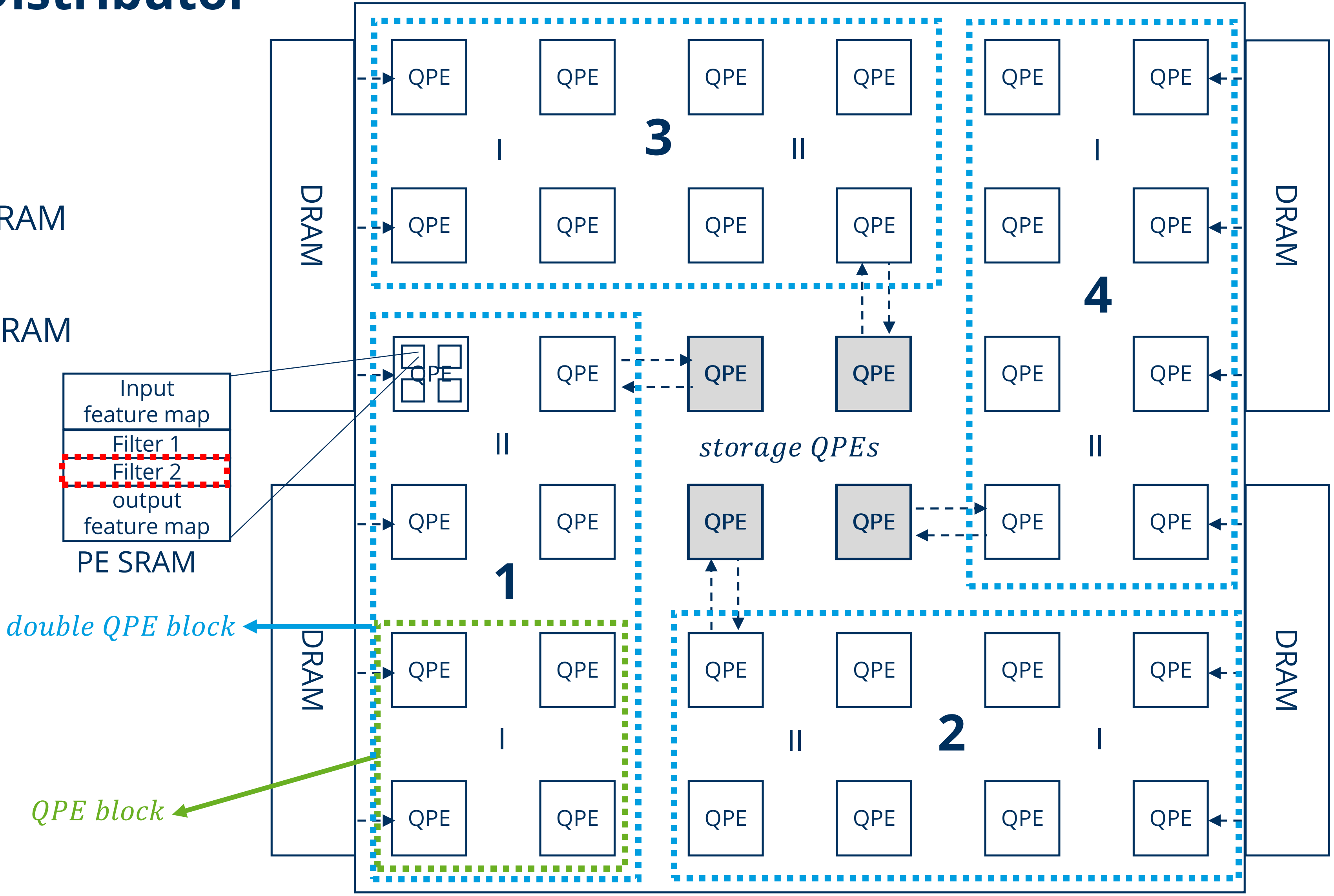


Mapping Strategy: Distributor

Data reuse in SpiNNaker2

- Storage QPE
→ decrease bottleneck caused by DRAM

- Data migration to reuse data
→ decrease bottleneck caused by DRAM



Only feature map reuse is available for SpiNNaker2!!

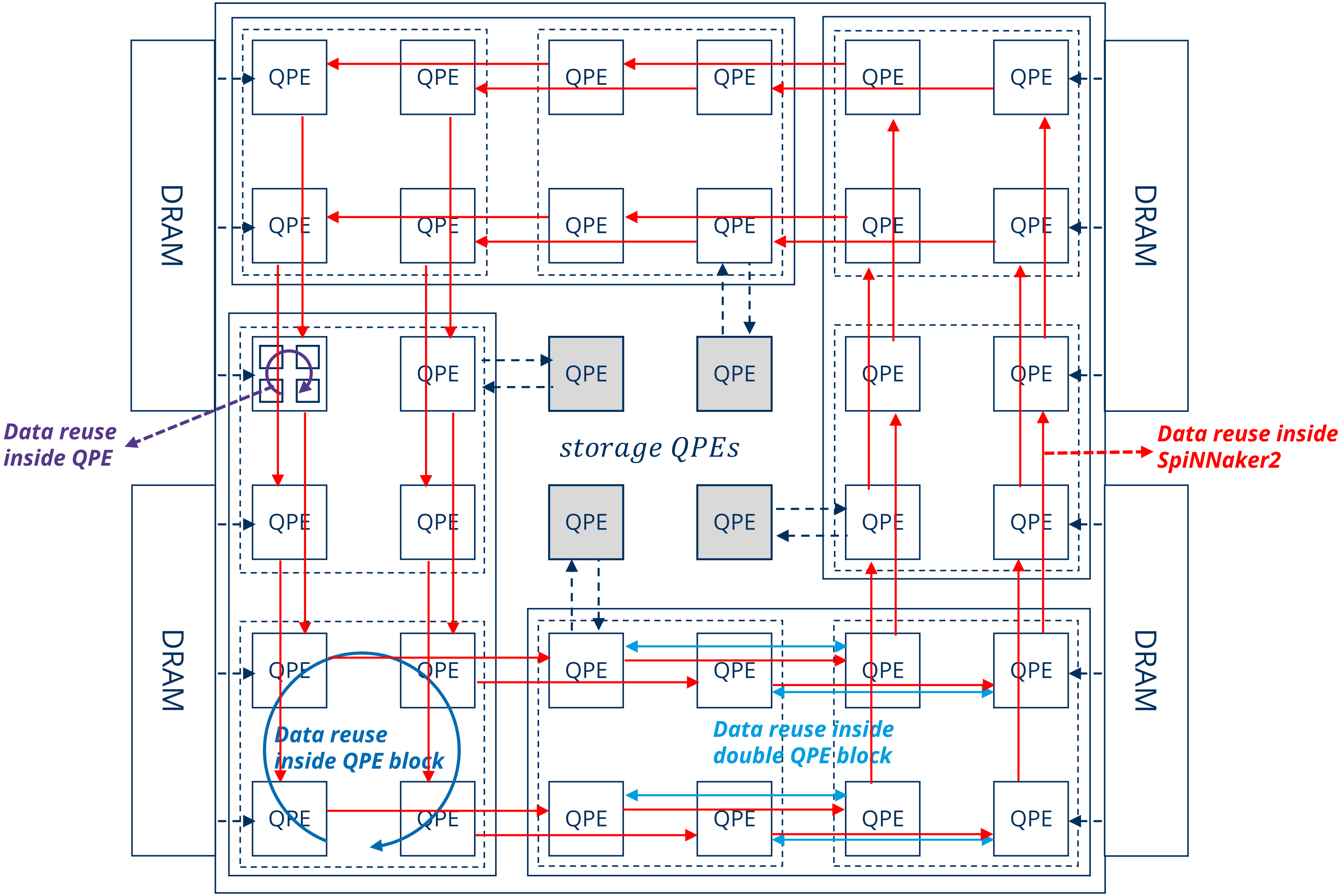
Mapping Strategy: Distributor

Data reuse in SpiNNaker2

4 ways to reuse data through **data migration**

- *Data reuse inside QPE*
- ***Data reuse inside QPE block***
- ***Data reuse inside double QPE block***
- ***Data reuse inside SpiNNaker2***

Only feature map reuse is available for SpiNNaker2!!



Content

Validation and Simulation

Validation: Splitter and QPE Simulator

- The split scheme will be verified (PASSED)

- Because all the simulation work is done on SpiNNaker2Py
  → Verification of the accuracy of QPE simulator.

  QPE clock cycles compared between simulator and ICPRO for various layers and local/neighbor weight. The difference is below 10%. $|\delta| \leq 10\%$

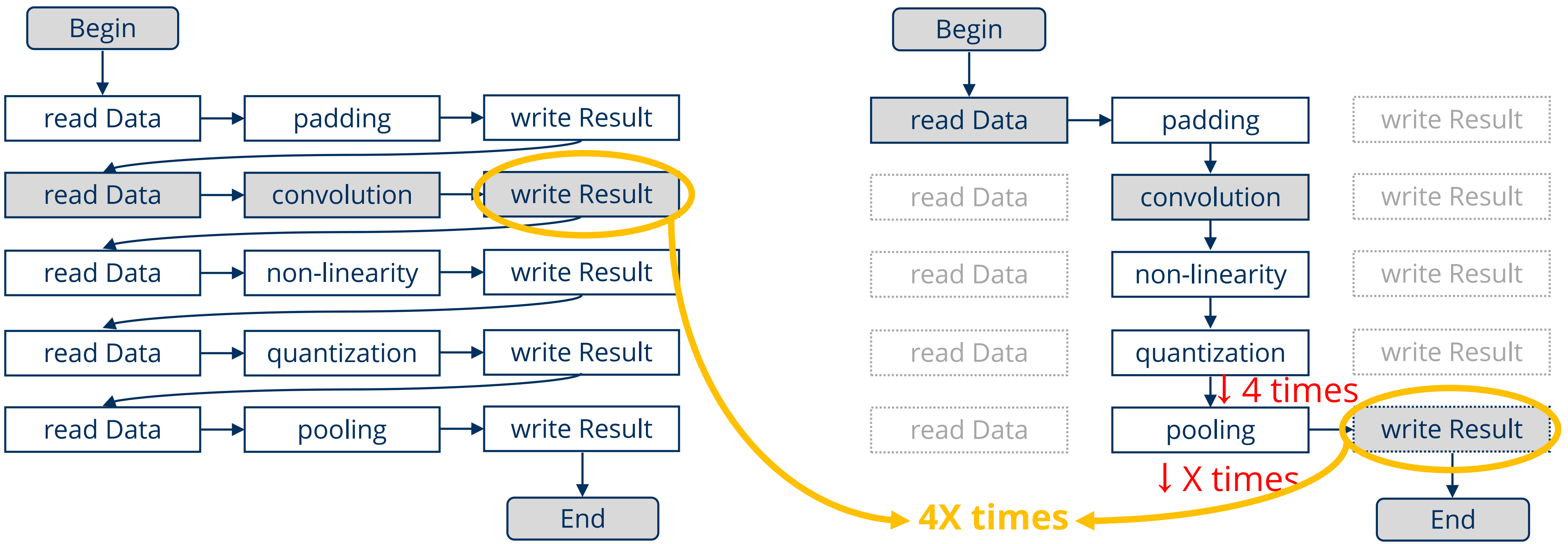
  Clocks deviation:

$$\delta = \frac{CLK_{simulator} - CLK_{ICPRO}}{CLK_{ICPRO}}$$

Validation and Simulation

Simulation:

- 3 distribution strategies for convolution and matrix-multiplication (**grey parts**)

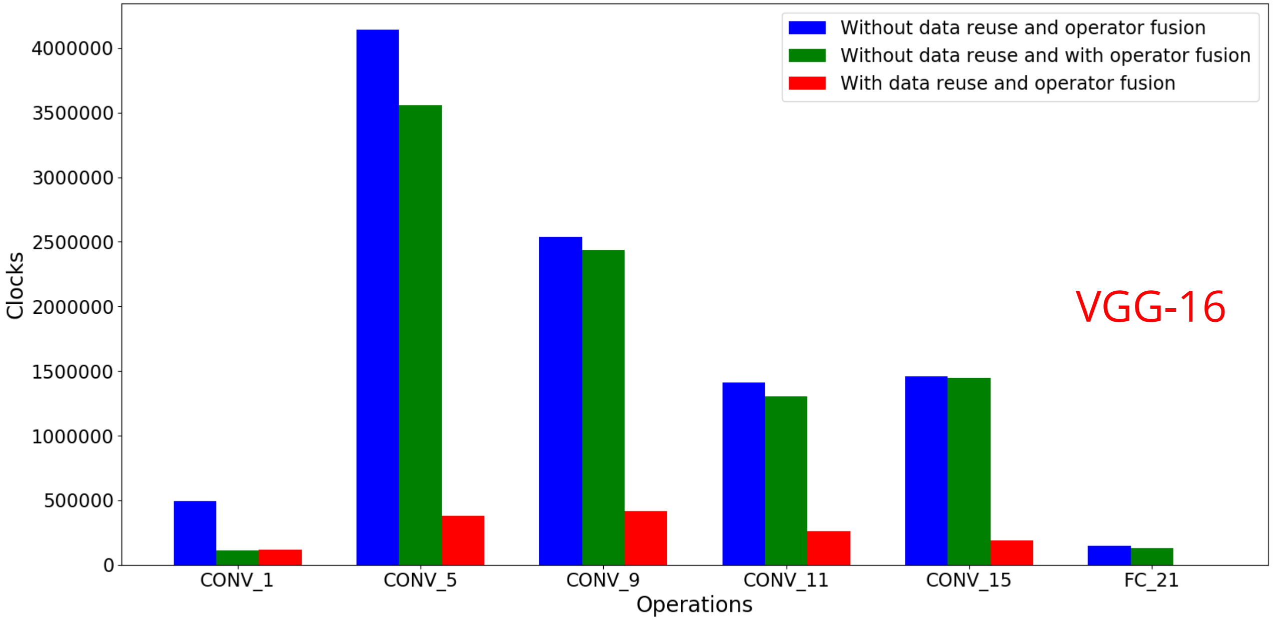


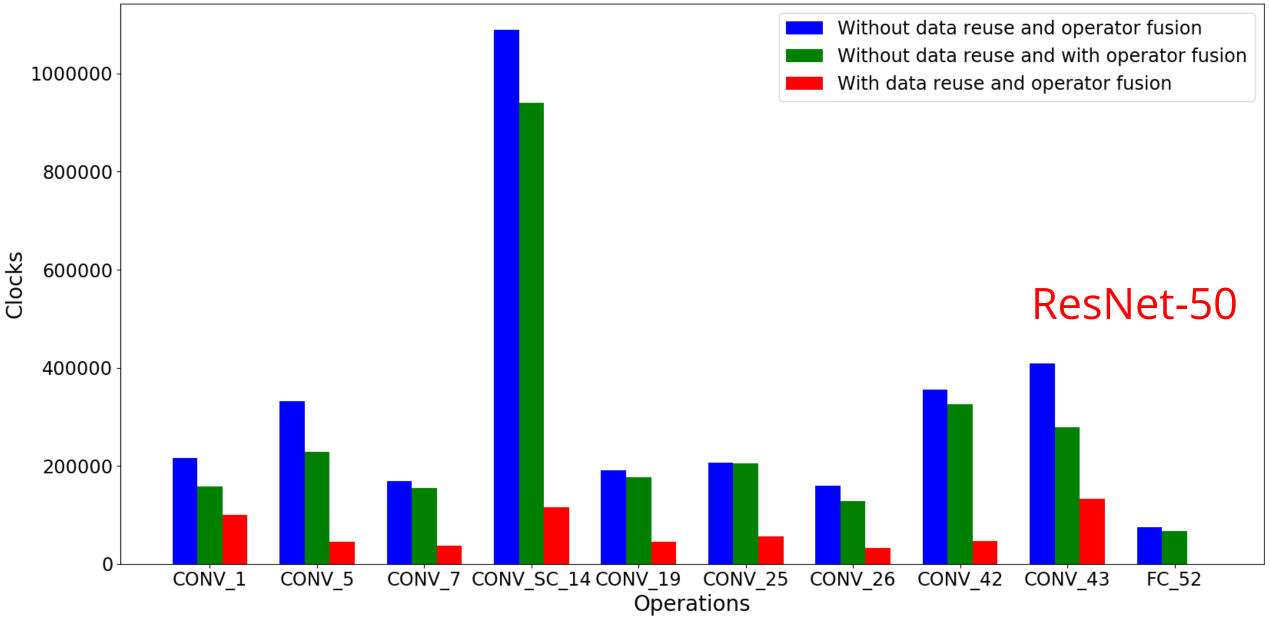
Without operator fusion (convolution block)

With operator fusion (convolution block)

Validation and Simulation

Simulation: 3 distribution strategies on SpiNNaker2





Comparing to without operator fusion and data reuse:

Operator fusion has an improvement **up to 5 times**. But most of them are below 2.

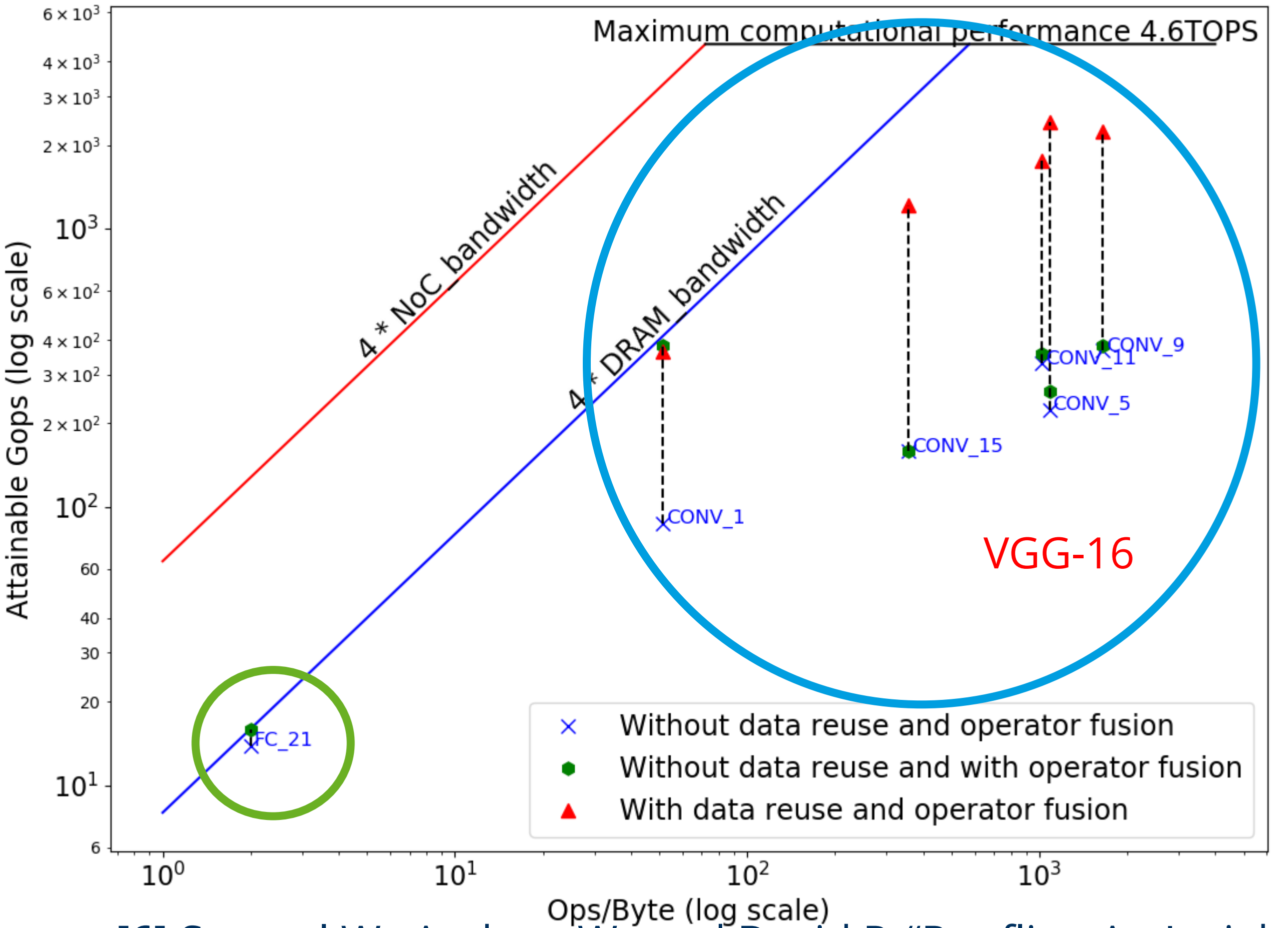
Data reuse + operator fusion has an improvement **up to 10 times** (**5~10 times**).

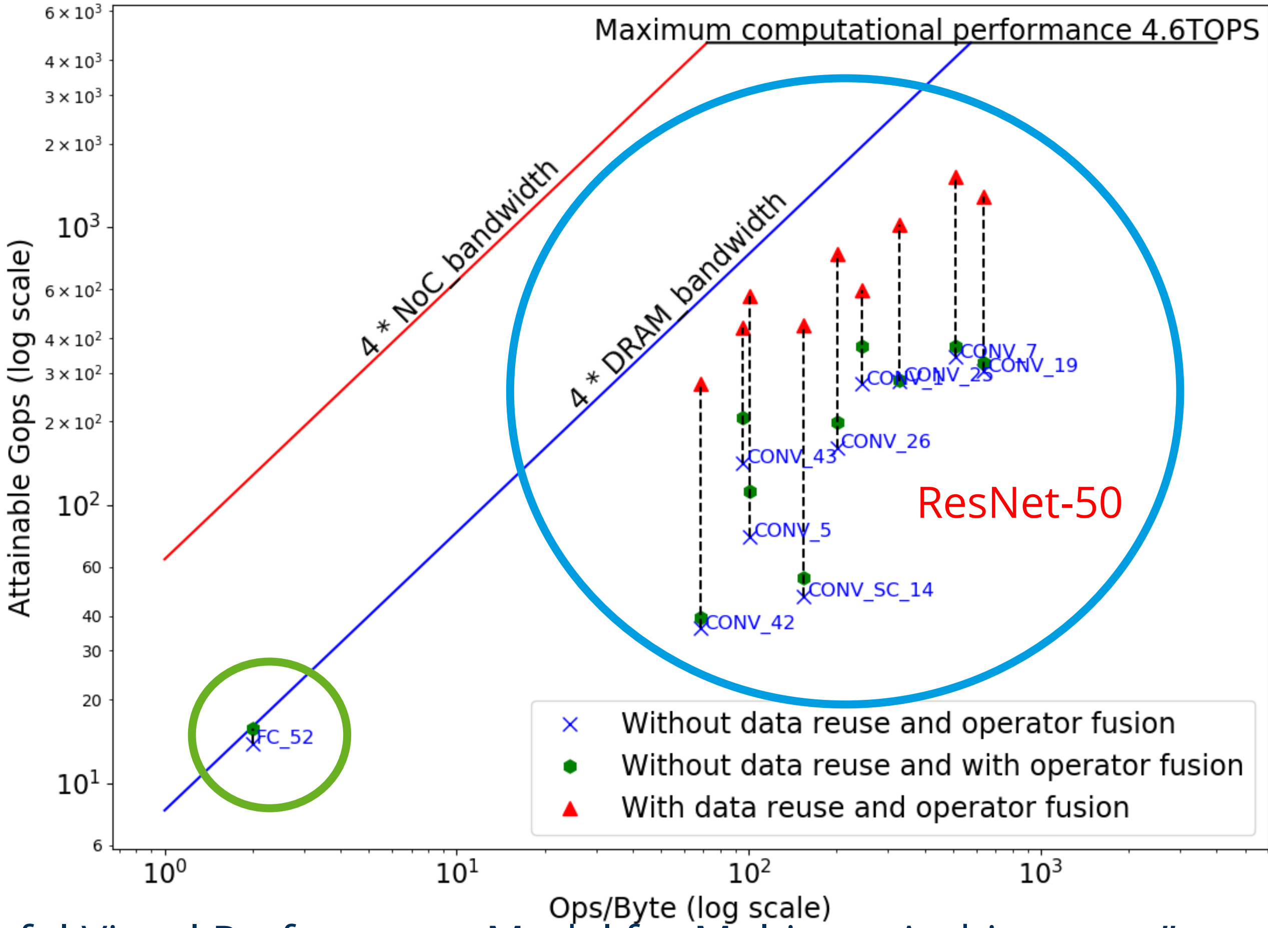
→ Operator fusion and data reuse can improve the performance. Data reuse helps much more!

Validation and Simulation

- MM: easily reaches the DRAM bandwidth ceiling.
- CONV: towards SpiNNaker2 performance ceiling

Simulation: 3 distribution strategies on SpiNNaker2

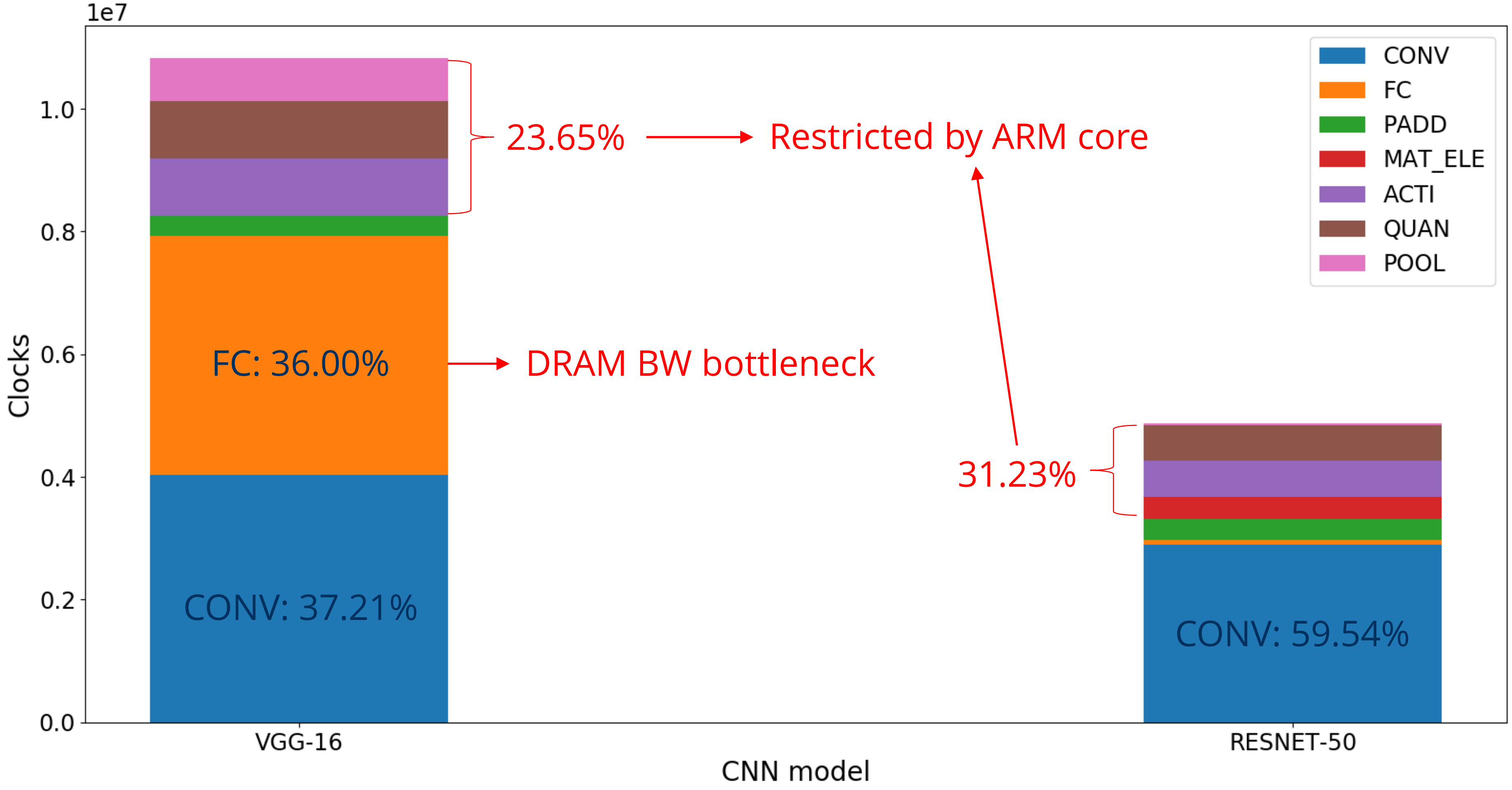




[6] Samuel W., Andrew W., and David P. “Roofline:An Insightful Visual Performance Model for Multicore Architectures”

Validation and Simulation

Simulation: Overall clocks of the whole network, SpiNNaker2, data reuse and operator fusion



VGG-16: ~43.3 ms

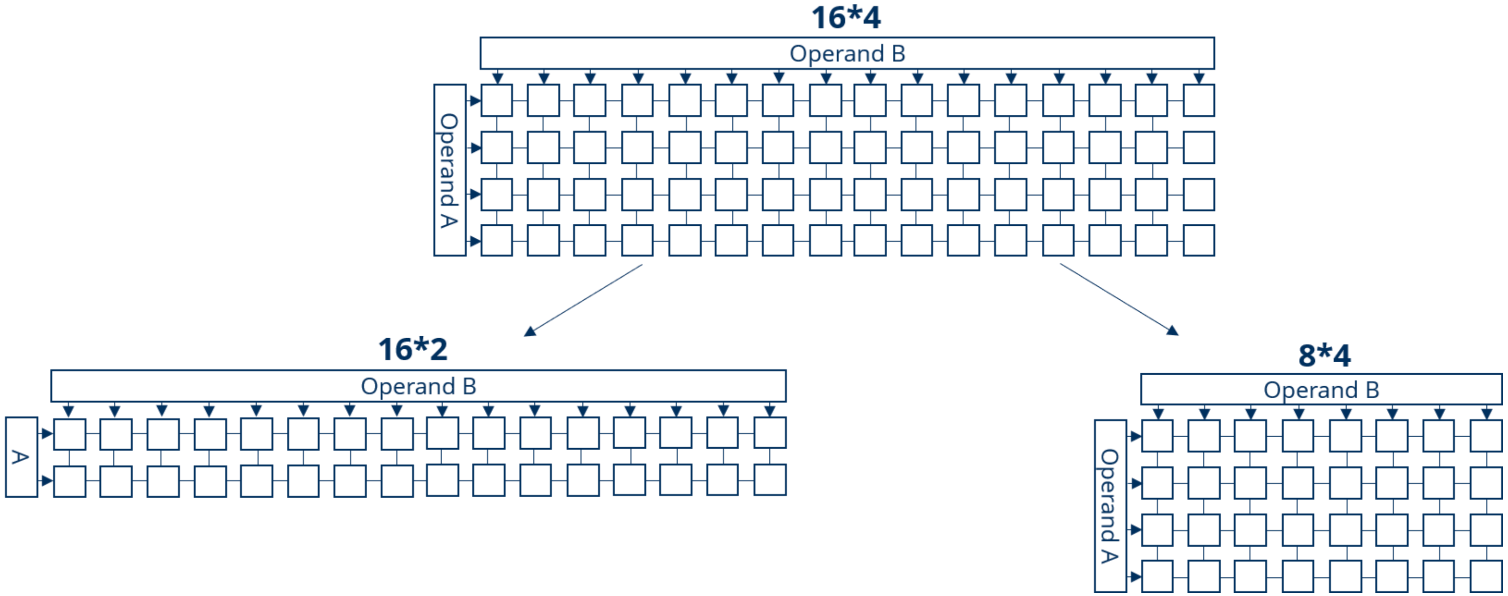
ResNet-50: ~19.5 ms

Validation and Simulation

Simulation: Comparison of MLAs with different number of MAC units

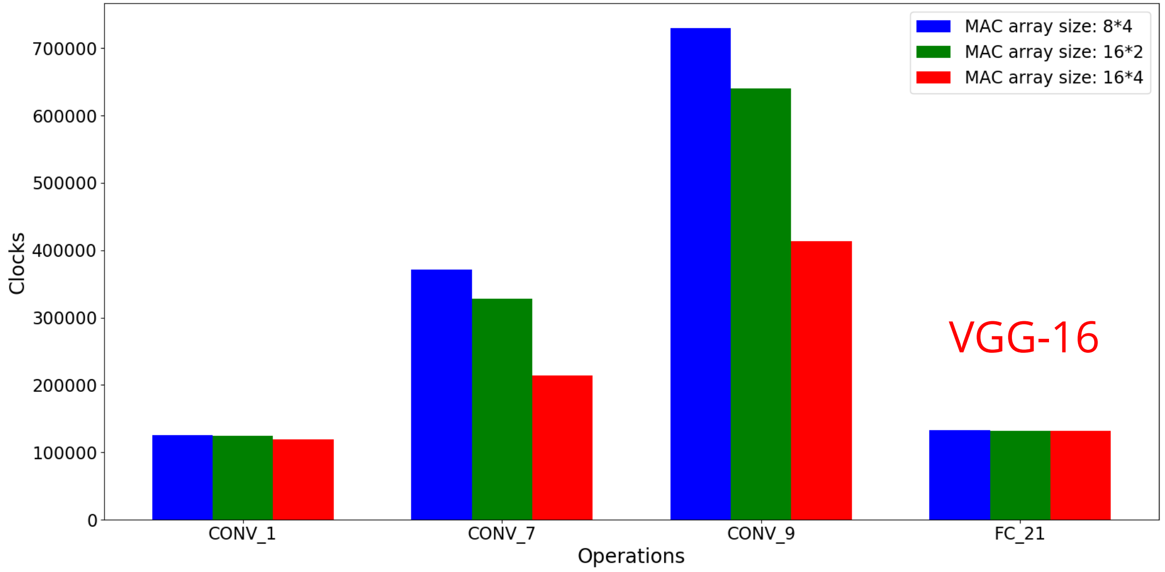
- The later proposed CNN (ResNet-50) has lower operational intensity
  →The computing resource cannot be fully utilize

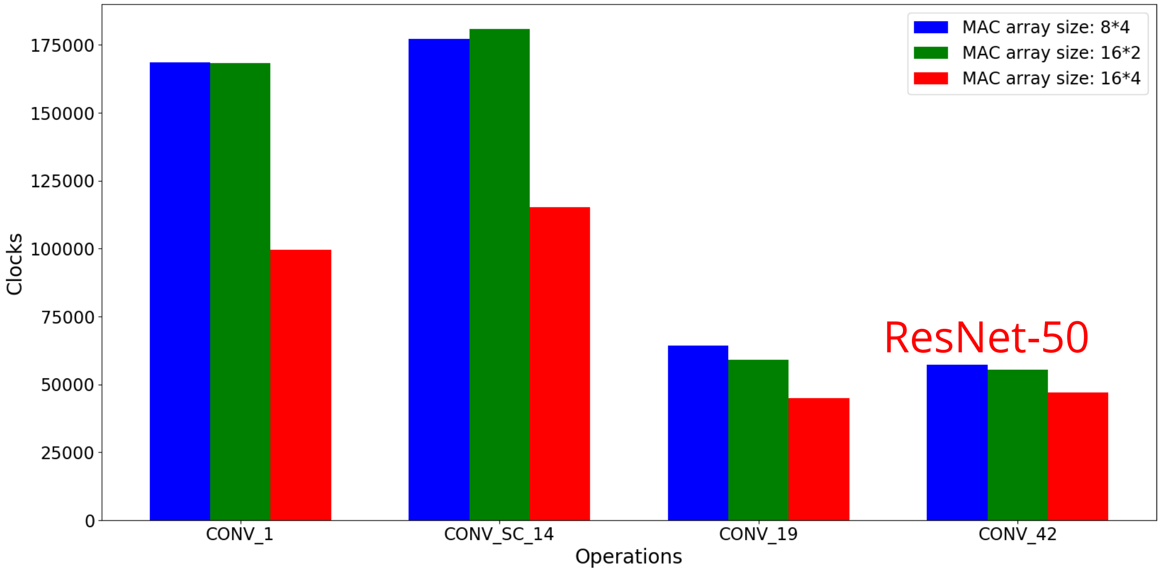
- Might decrease the chip area and power consumption



Validation and Simulation

Simulation: Comparison between MLAs with different number of MAC units





- Computing power is halved, but the degradation is below 1.5 times
  → alleviate the problem of insufficient memory bandwidth

- 16*2 is better than 8*4
  → 16*2 has less data fetching operations.

Content

- Motivation
- SpiNNaker2 and Simulator: SpiNNaker2Py
- Mapping Strategy
- Validation and Simulation
- Conclusion

Conclusion

- Contributions :
  1. SpiNNaker2 Simulator: SpiNNaker2Py ;
  2. By optimized split scheme, operator fusion and several hierarchies of data reuse, the achieved speedup on SpiNNaker2 is up to 10;
  3. The system is limited by memory bandwidth;
  4. comparison of different MLA architectures;

- Improvements:
  1. improvement the simulation speed of SpiNNaker2Py
  2. machine learning based search algorithm for splitter
  3. Distributor also has room for improvement through pre-caching

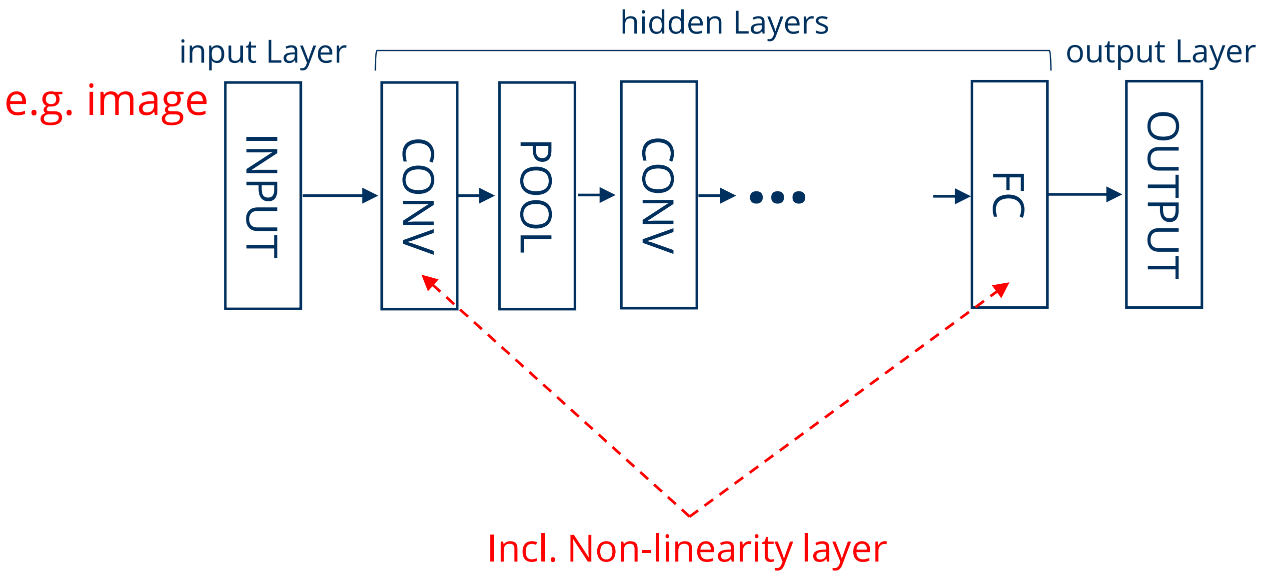
Reference

1. Sebastian Hoeppner and Christian Mayr. SpiNNaker2 Towards extremely efficient digital neuromorphics and multi-scale brain emulation. 2018
2. TU Dresden. SpiNNaker2 Wiki: SpiNNaker2 Universal Spiking Neural Network Architecture
3. Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 2015. url: http://arxiv.org/abs/1409.1556
4. K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016, pp. 770–778. doi: 10.1109/ CVPR.2016.90
5. T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L Ceze, C. Guestrin, and A. Krishnamurthy. "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning". In: (2018). url: https://arxiv.org/abs/1802.04799
6. Samuel Williams, Andrew Waterman, and David Patterson. "Roofline:An Insightful Visual Performance Model for Multicore Architectures".In: Commun. ACM 52.4 (Apr. 2009), pp. 65–76. issn: 0001-0782. doi:10.1145/1498765.1498785. url: http://doi.acm.org/10.1145/1498765.1498785

Thank you

Mapping Strategy:

CNN architecture:



CONV: convolutional layer
POOL: pooling layer
FC: fully-connected layer

Mapping Strategy: Parser

Layer → Operations: primitive operations supported by SpiNNaker2

Type of Layers	Operations
$convolutional\ layer$	$padding\ operation\ (ARM)$
	$convolution\ operation\ (MLA)$
	$nonlinearity\ operation\ (ARM)$
	$quantization\ operation\ (ARM)$
$pooling\ layer$	$padding\ operation\ (ARM)$
	$pooling\ operation\ (MLA/ARM)$
$fully-connected\ layer$	$matrix\ multiplication\ operation\ (MLA)$
	$nonlinearity\ operation\ (ARM)$
	$quantization\ operation\ (ARM)$

Mapping Strategy: Splitter

Convolution block

- The Core operation “convolution operation” is the split object.
- Split dimension order:

$$channel_{out} \rightarrow width_{out}, height_{out} \rightarrow channel_{in}$$

- If $channel_{in}$ is split:

$$\begin{bmatrix} padding\ operation\ (ARM) \\ \boldsymbol{convolution\ operation(MLA)} \\ element - wise\ addition(ARM) \\ nonlinearity\ operation(ARM) \\ quantization\ operation(ARM) \end{bmatrix} \longrightarrow convolution\ block$$

- SRAM utilization, MAC utilization, PE utilization, size increasement, computation balance are considered during splitting.

Mapping Strategy: Splitter

Pooling block

- The Core operation “pooling operation” is the split object.
- Split dimension order:

$$channel \rightarrow width_{out}, height_{out}$$

- SRAM utilization, MAC utilization, PE utilization, size increasement are considered during splitting.

Mapping Strategy: Splitter

Matrix multiplication block

- The Core operation “matrix multiplication operation” is the split object.

- Split dimension order:

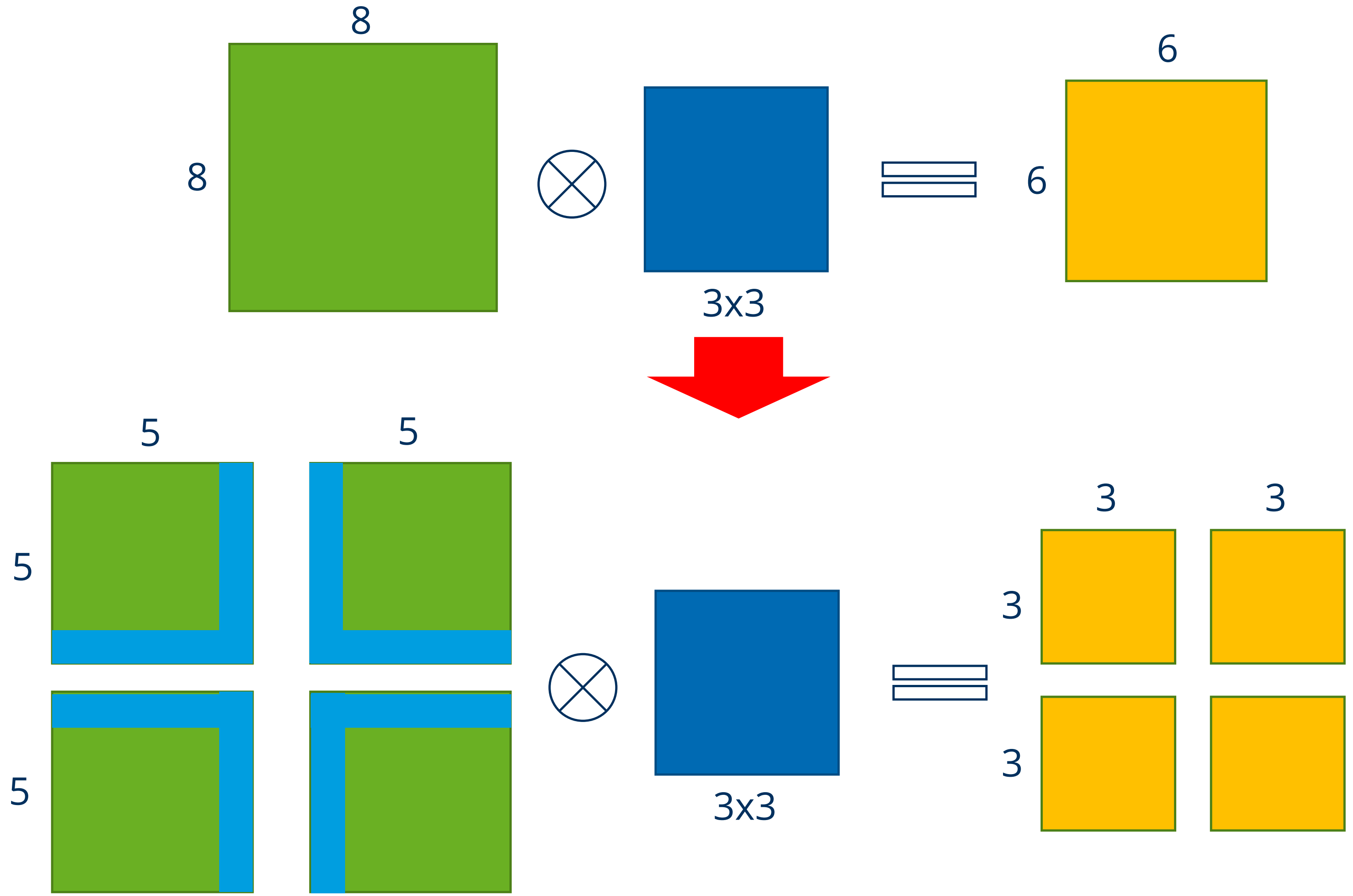
$$height_{weight} \rightarrow width_{weight}$$

- If $height_{weight}$ is split:

$$\begin{bmatrix} \boldsymbol{matrix\ multiplication\ operation(MLA)} \\ {\color{red} element - wise\ addition(ARM)} \\ nonlinearity\ operation\ (ARM) \\ quantization\ opoeration\ (ARM) \end{bmatrix} \longrightarrow matrix\ multiplication\ block$$

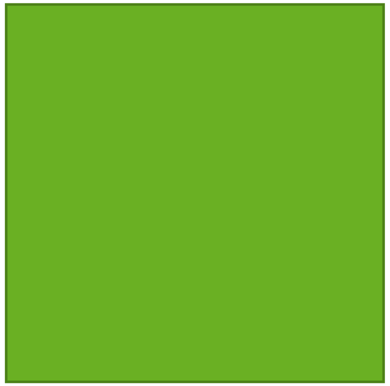
Mapping Strategy: Splitter

Convolution block: convolution



Mapping Strategy: Splitter

Convolution block: convolution



Input feature map

Split input feature map into C parts

- W → w parts
- H → h parts
- $C = w * h$

$$Size_{before} = H * W$$

$$\begin{aligned} Size_{after} = H * W \\ +(h-1) * W * (H_{filter} - S) \\ +(w-1) * H * (W_{filter} - S) \\ -(h-1) * (w-1) * (H_{filter} - S) * (W_{filter} - S) \end{aligned}$$

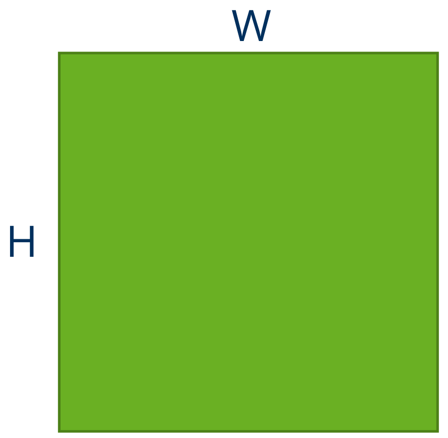
H_{filter}: height of filter weight

W_{filter}: width of filter weight

S: stride

Mapping Strategy: Splitter

Convolution block: convolution



Input feature map

$$
\begin{aligned}
Size_{increased} &= Size_{after} - Size_{before} \\
&= (h-1) * W * \left(H_{filter} - S\right) \\
&\quad +(w-1) * H * \left(W_{filter} - S\right) \\
&\quad -(h-1) * (w-1) * \left(H_{filter} - S\right) * \left(W_{filter} - S\right)
\end{aligned}
$$

$W_{filter} = H_{filter} = F;$
$W = H;$
$C = w * h$

$$Size_{increased} = \left(\frac{C}{w} + w - 2\right) * H * (F - S) - \left(\frac{C}{w} - 1\right) * (w-1) * (F-S)^2$$

Setting the gradient of $Size_{increased}$ with respect to w to be zero

$$w - \frac{C}{w} = 0$$

$$w = \sqrt{C}, h = \sqrt{C}$$

Mapping Strategy: Splitter

Convolution block

Lead some calculations to be accelerated by ARM Core

$$input\ feature\ map: [\boxed{width_{in}, height_{in}}, \boxed{channel_{in}}]$$

$$filter\ weight: [width_{filter}, height_{filter}, \boxed{channel_{in}}, \boxed{channel_{out}}]$$

$$output\ feature\ map: [\boxed{width_{out}, height_{out}}, \boxed{channel_{out}}]$$

Width need to align to
MAC_ARRAY_COLUMN (16)

Alignment with
MAC_ARRAY_ROW (4)

Mapping Strategy: Splitter

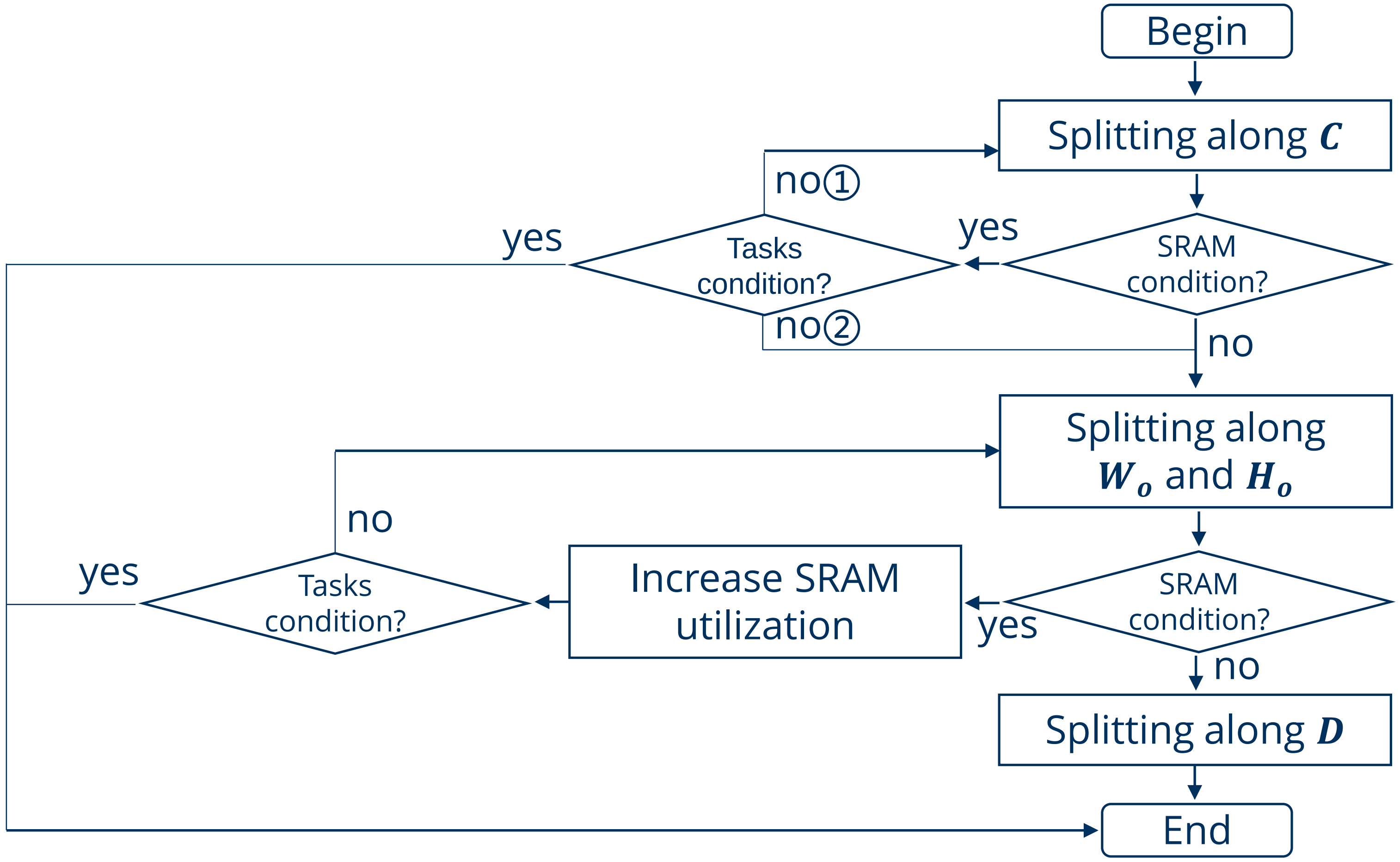
Convolution block

SRAM condition: If the available SRAM is enough for input, weight and output?

Tasks condition: If the number of the split tasks is
≥ 4 for QPE or
≥ 128 for SpiNNaker2?

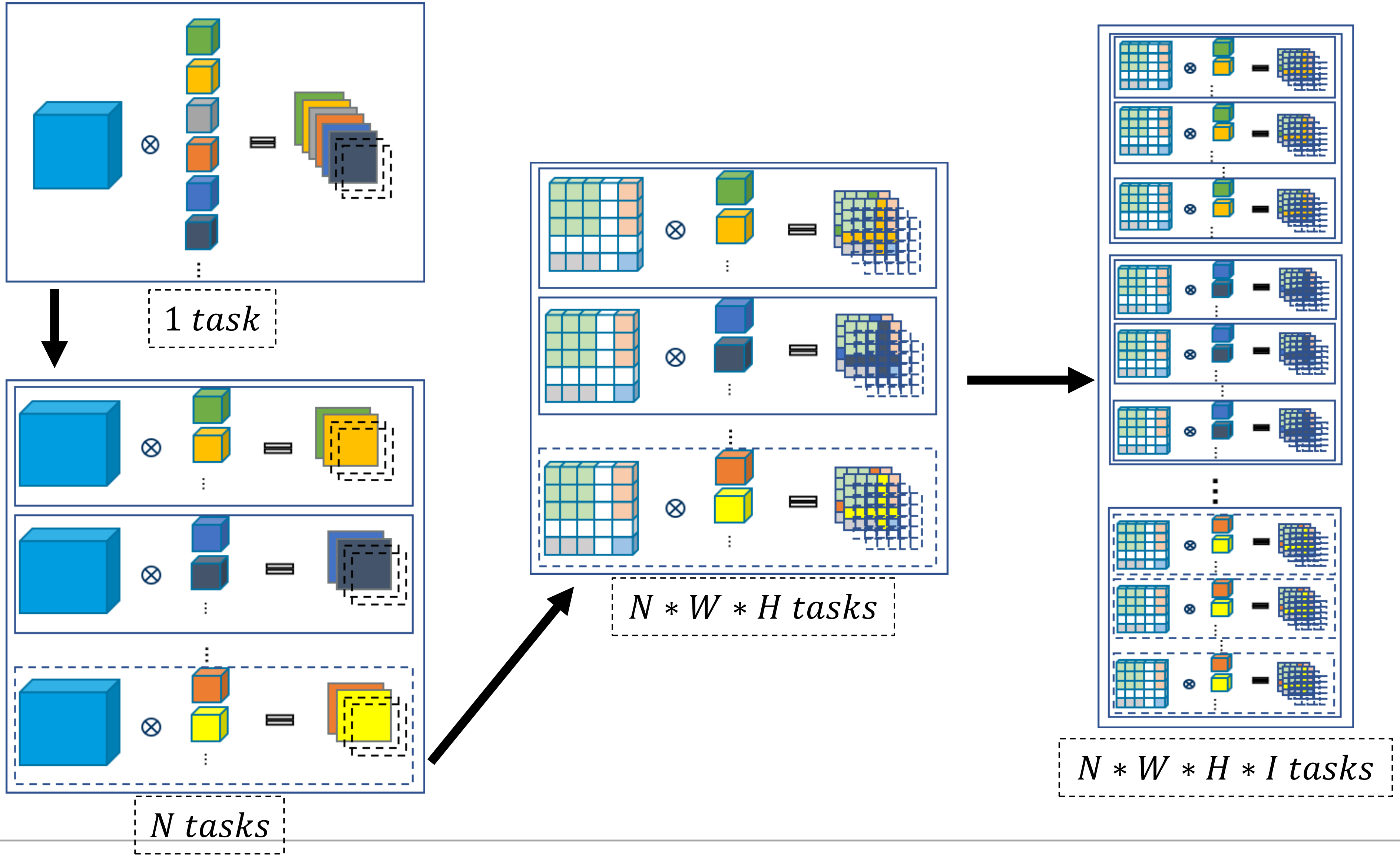
no①: The number of tasks can be increased by splitting C into more parts.

no②: The number of tasks cannot be increased by splitting C into more parts.



Mapping Strategy: Splitter

Convolution



Mapping Strategy: Splitter

Pooling block

Lead some calculations to be accelerated by ARM Core

$$input\ feature\ map: [\boxed{width_{in}, height_{in}}, \boxed{channel]}$$

$$output\ feature\ map: [\boxed{width_{out}, height_{out}}, \boxed{channel]}$$

Width need to align to **MAC_ARRAY_COLUMN (16)**

Alignment with **MAC_ARRAY_ROW (4)**

Mapping Strategy: Splitter

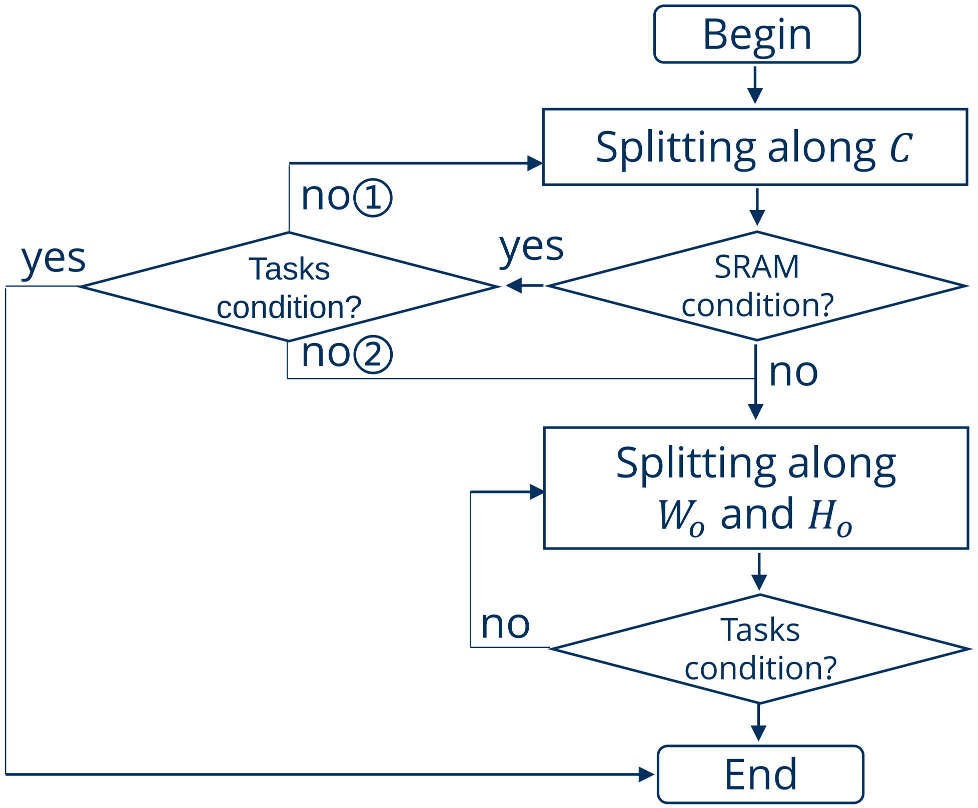
Pooling block

SRAM condition: If the available SRAM is enough for input, weight and output?

Tasks condition: If the number of the split tasks is
≥ 4 for QPE or
≥ 128 for SpiNNaker2?

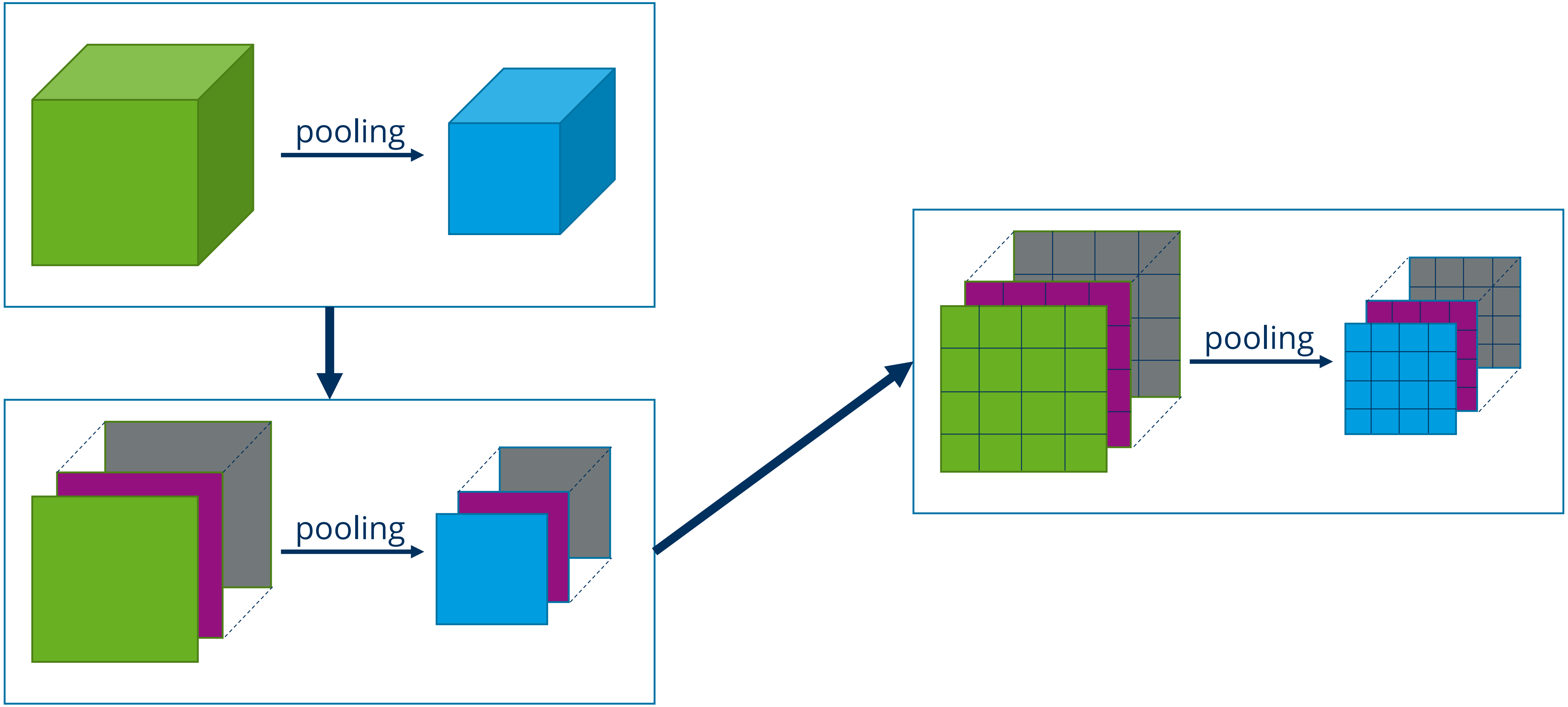
no①: The number of tasks can be increased by splitting C into more parts.

no②: The number of tasks cannot be increased by splitting C into more parts.



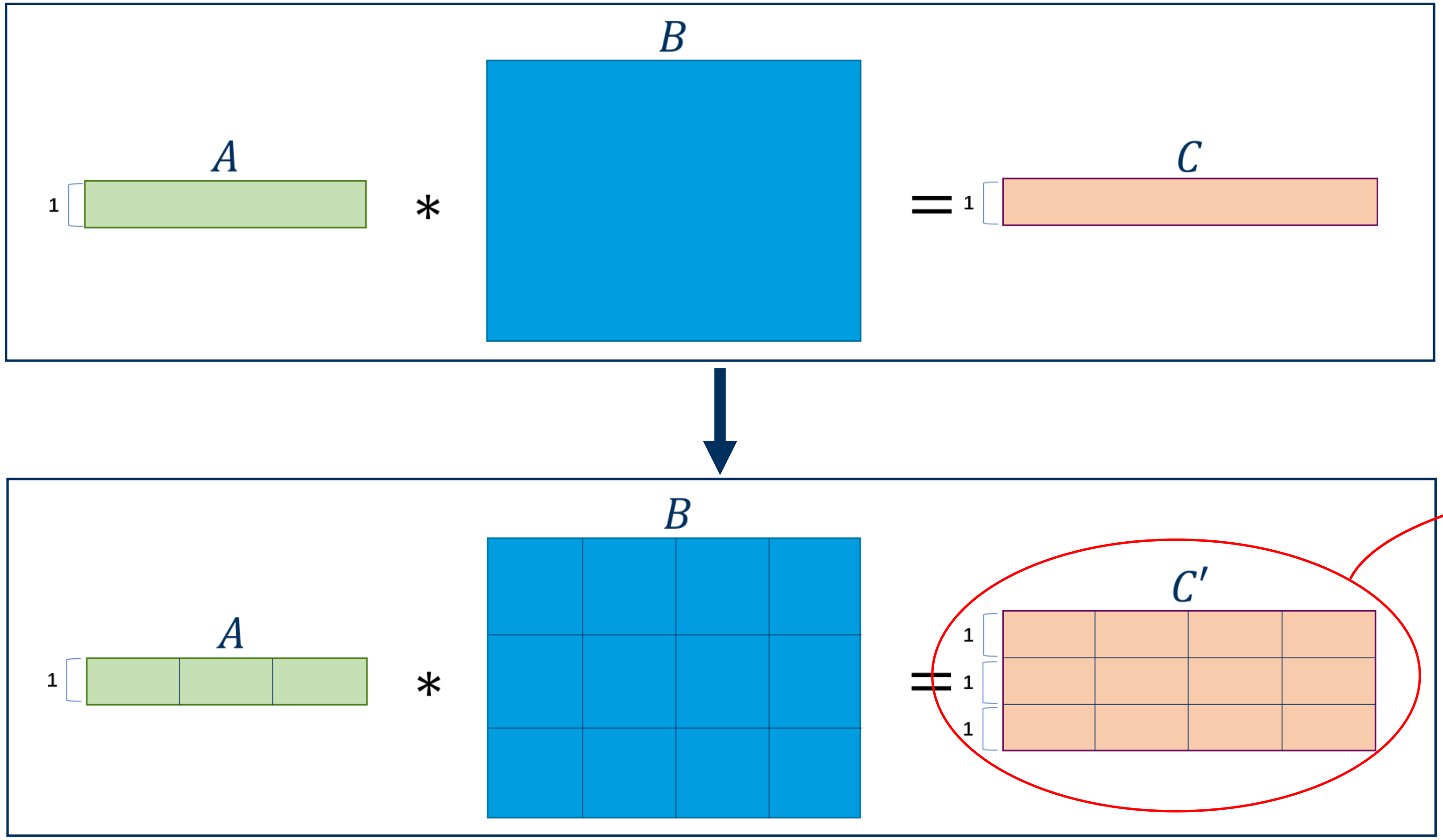
Mapping Strategy: Splitter

Pooling block

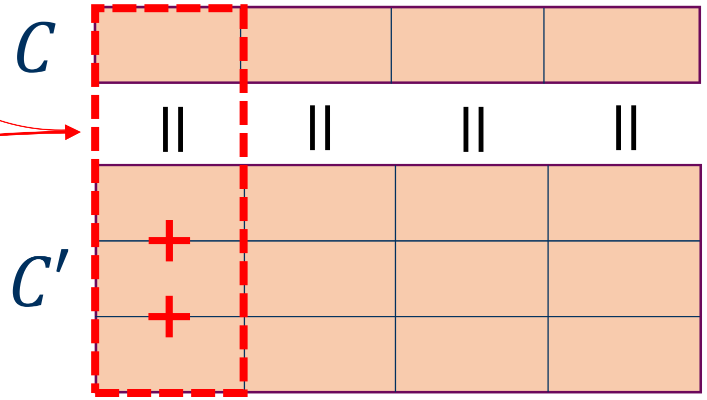


Mapping Strategy: Splitter

Matrix Multiplication block



$$\begin{array}{l} matrix\ multiplication\ block \\ \left[\begin{array}{l} \boldsymbol{matrix\ multiplication\ operation}(\boldsymbol{MLA}) \\ element-wise\ addition\ (MLA) \\ nonlinearity\ operation\ (ARM) \\ quantization\ operation\ (ARM) \end{array}\right] \end{array}$$



Mapping Strategy: Splitter

Matrix Multiplication block: MM

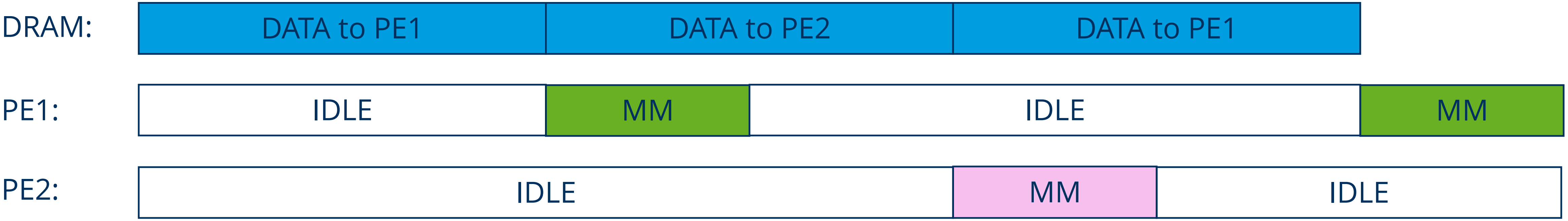
$A * B = C$, with dimension $[W_A, H_A]$ and $[W_B, H_B]$

Read matrix B from DRAM takes (comparing to matrix B, matrix A is very small)

$$T_{DRAM} = \frac{W_B * H_B}{f_{DRAM} * 16\ Bytes\ /\ 2}$$

The computation taskes

$$T_{computation} = \frac{W_B * H_B * 2}{f_{MLA} * 16 * 2} = \frac{W_B * H_B}{f_{MLA} * 16} = \frac{1}{2} T_{DRAM}$$



Validation, Simulation and Experiment

Validation: QPE Simulator

CONV/FC: local PE SRAM

Clocks deviation:

$$\delta = \frac{CLK_{simulator} - CLK_{ICPRO}}{CLK_{ICPRO}}$$

$\delta \in [-7.12\%, 5.04\%]$

→ meet the requirement $|\delta| \leq 10\%$

Convolution task feature map dimension: $[W, H, D]$, filter dimension: $[W_f, H_f, D, C]$, stride: 1	**Clocks** (HDL prototype)	**Clocks** (Simulator)	**Clock deviation δ** $\left(\frac{CLK_{simulator} - CLK_{ICPRO}}{CLK_{ICPRO}}\right)$
fmap: [226,22,3] filter: [3,3,3,4]	27748	25771	-7.12%
fmap: [114,9,64] filter: [3,3,64,4]	61186	59543	-2.69%
fmap: [18,18,128] filter: [3,3,128,4]	38626	38264	-0.94%
fmap: [30,9,256] filter: [3,3,256,4]	66244	66342	0.15%
fmap: [56,14,64] filter: [1,1,64,4]	10822	10599	-2.06%
fmap: [28,10,256] filter: [1,1,256,4]	13162	13395	1.77%
fmap: [28,14,128] filter: [5,5,128,4]	116215	109402	-5.86%
fmap: [28,10,128] filter: [7,7,128,4]	82139	86275	5.04%
fmap: [16,16,128] filter: [9,9,128,4]	31648	32883	3.90%

Matrix Multiplication task Matrix **A** dimension: $[W_A, H_A]$, Matrix **B** dimension: $[W_B, H_B]$	**Clocks** (HDL prototype)	**Clocks** (Simulator)	**Clock deviation δ** $\left(\frac{CLK_{simulator} - CLK_{ICPRO}}{CLK_{ICPRO}}\right)$
fmap: [64,1] weight: [1024, 64]	13276	12619	-4.95%
fmap: [128,1] weight: [512,128]	11908	11435	-3.97%

Validation, Simulation and Experiment

Validation: QPE Simulator

<u>CONV/FC: neighbor PE SRAM</u>

Clock deviation:

$$\delta = \frac{CLK_{simulator} - CLK_{ICPRO}}{CLK_{ICPRO}}$$

$\delta \in [-9.51\%, -0.80\%]$

→ meet the requirement $|\delta| \leq 10\%$

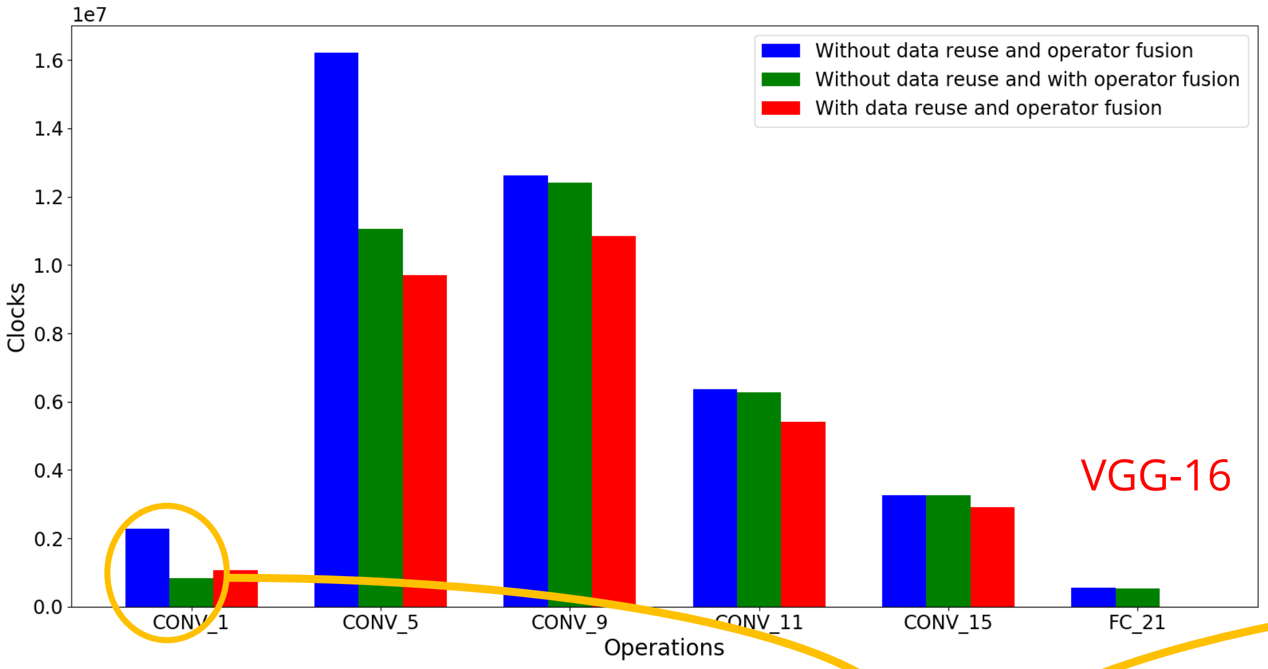
Because of design flaws of HDL, only one task is available.

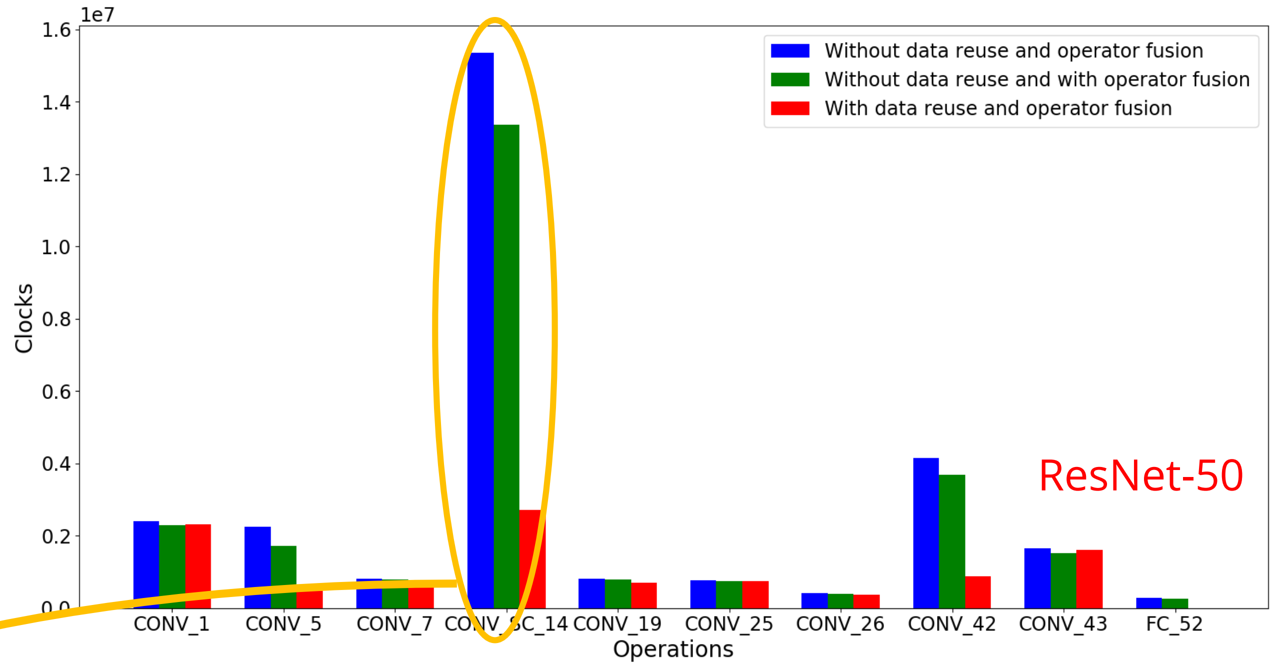
Neighbor PE SRAM feature map dimension: $[W, H, D] = [226,22,3]$, filter dimension: $[W_f, H_f, D, C] = [3,3,3,4]$, stride: 1	**Clocks (Error to shift 0)** (HDL prototype)	**Clocks (Error to shift 0)** (Simulator)	**Clock deviation** $(\frac{CLK_{simulator} - CLK_{ICPRO}}{CLK_{ICPRO}})$
Neighbor PE shift: 0	27748 (0.00%)	25771 (0.00%)	-7.12%
Neighbor PE shift: 1	27735 (0.00%)	25772 (0.00%)	-7.08%
Neighbor PE shift: 2	28482 (2.65%)	25772 (0.00%)	-9.51%
Neighbor PE shift: 3	27726 (0.00%)	25773 (0.00%)	-7.04%

Neighbor PE SRAM Matrix **A** dimension: $[W_A, H_A] = [64, 1]$, Matrix **B** dimension: $[W_B, H_B] = [1024, 64]$	**Clocks (Error to shift 0)** (HDL prototype)	**Clocks (Error to shift 0)** (Simulator)	**Clock deviation** $(\frac{CLK_{simulator} - CLK_{ICPRO}}{CLK_{ICPRO}})$
Neighbor PE shift: 0	13276 (0.00%)	12619 (0.00%)	-4.95%
Neighbor PE shift: 1	13563 (2.16%)	13454 (6.62%)	-0.80%
Neighbor PE shift: 2	12893 (2.88%)	12493 (-1.00%)	-3.10%
Neighbor PE shift: 3	13577 (2.27%)	12974 (2.81%)	-4.44%

Validation, Simulation and Experiment

Simulation: 3 distribution strategies for **QPE**





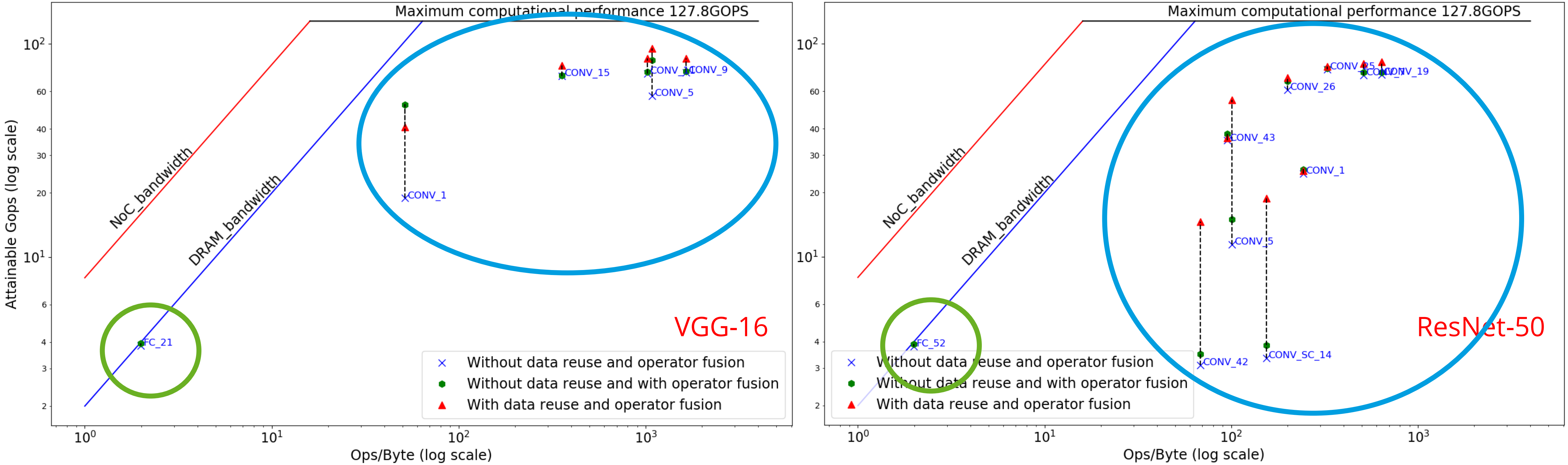
Green one has an improvement up to 3 times, comparing to blue one. <u>But most of them don't have that improvement.</u>

Red one has an improvement up to 5 times, comparing to blue one. <u>But most of them don't have that improvement.</u>

→ Operator fusion and data reuse can improve the performance.

Validation, Simulation and Experiment

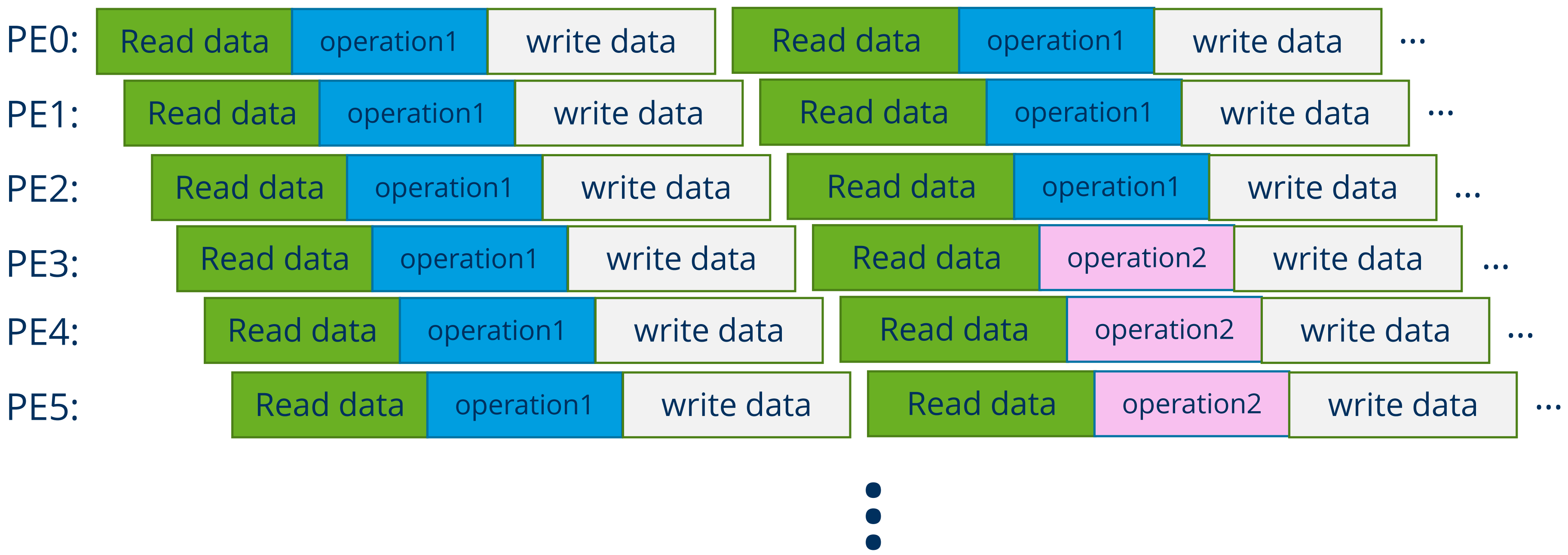
Simulation: 3 distribution strategies for <u>**QPE**</u>



- Matrix-multiplication: easily reaches the DRAM bandwidth ceiling.
- Convolution: Operator fusion and data reuse → towards QPE performance ceiling

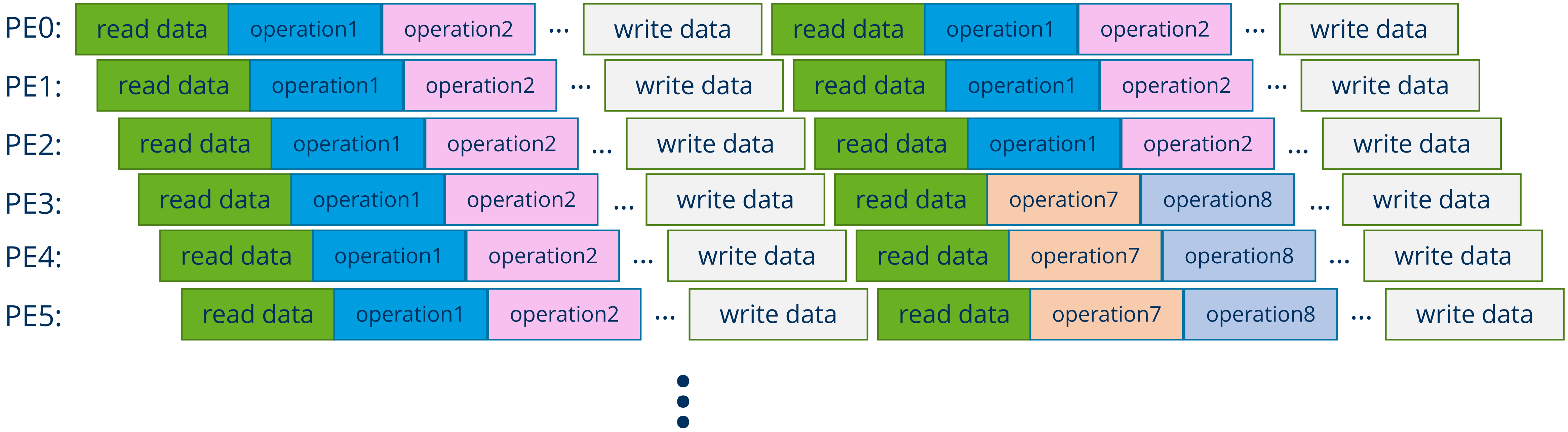
Conclusion: improvements on Distributor

Without operator fusion and data reuse



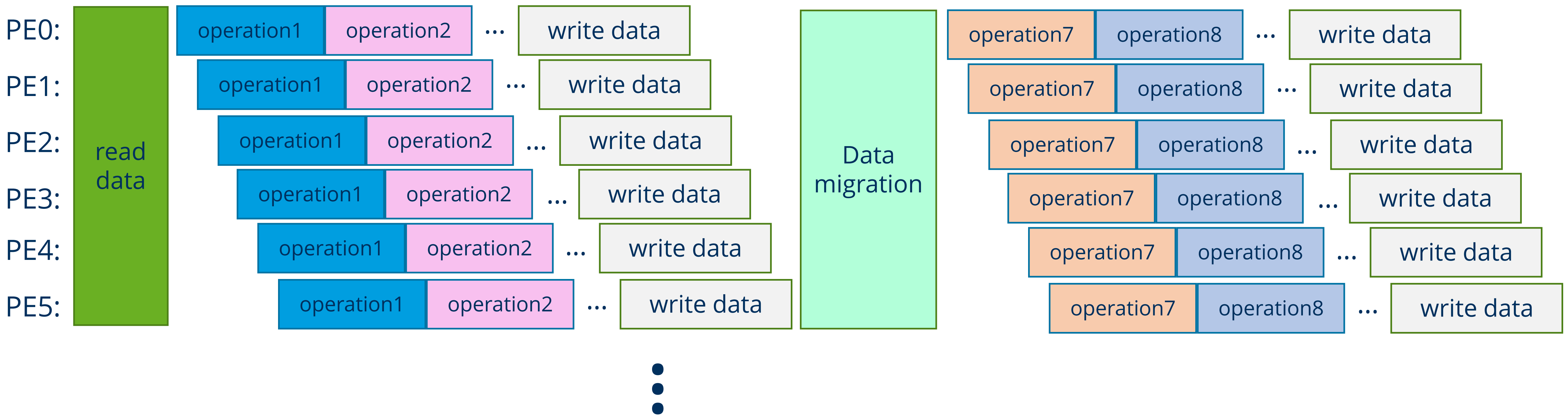
Conclusion: improvements on Distributor

With operator fusion and without data reuse



Conclusion: improvements on Distributor

With operator fusion and without data reuse



Validation, Simulation and Experiment

Simulation:

Components	Frequency (MHz)	Clocks per operation
NoC	500	1
DRAM	250	2
HOST Interface	250	1
PE	250	1
ARM in PE	250	1
SRAM in PE	250	1
DMA in PE	250	1

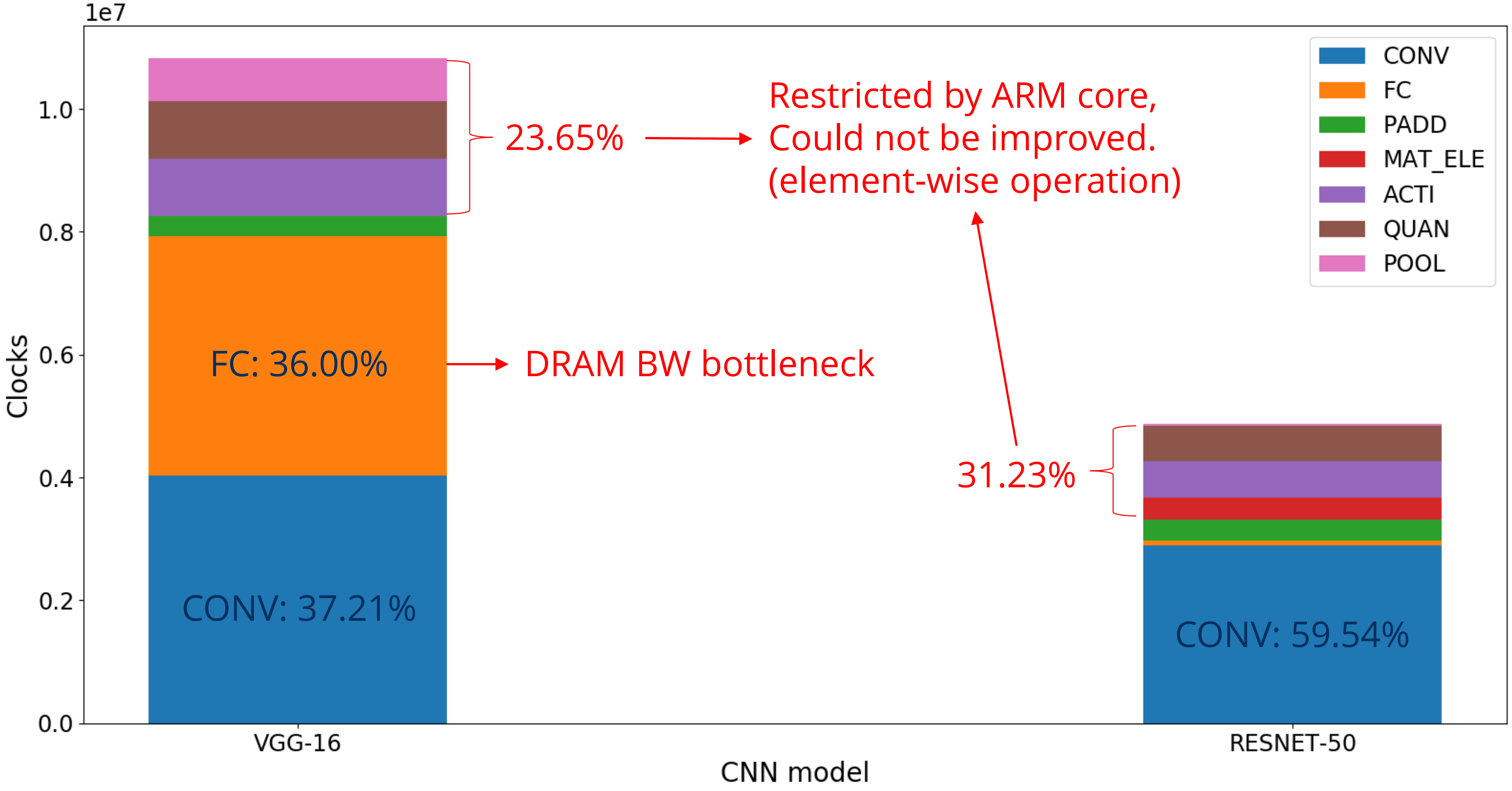
Validation, Simulation and Experiment

Simulation: Overall clocks of the whole network, SpiNNaker2, data reuse and operator fusion

Operation	Clocks	Comments
padding	2	per 32-bit
quantization	8	per input pixel
non-Linearity	8	32-bit ReLU, per input pixel
non-Linearity	2.5	8-bit ReLU, per input pixel
MAX-pooling	18.75	32-bit, per input pixel
MAX-pooling	12	8-bit, per input pixel
matrix element-wise addition	8	per input pixel

Validation and Simulation

Simulation: Overall clocks of the whole network, SpiNNaker2, data reuse and operator fusion

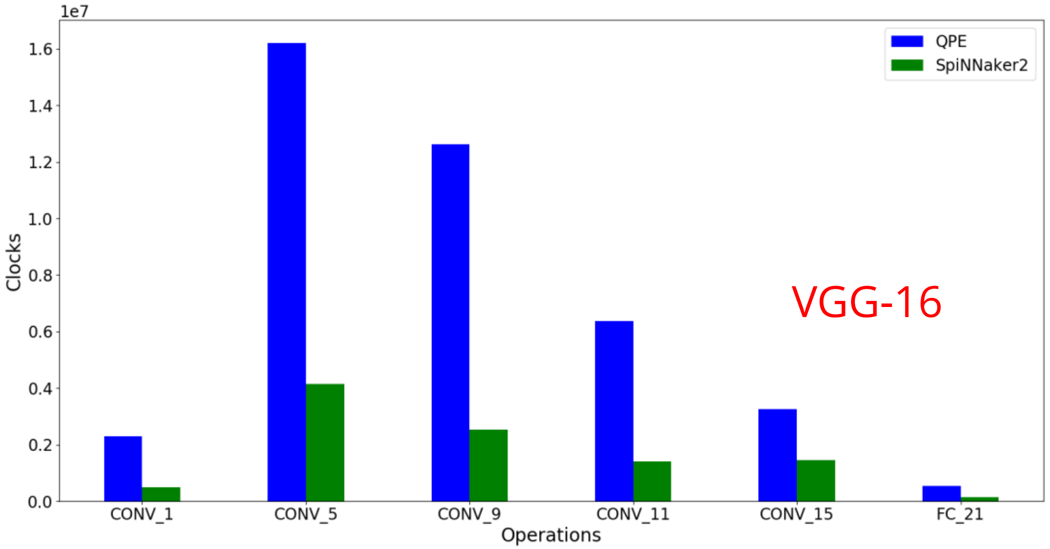


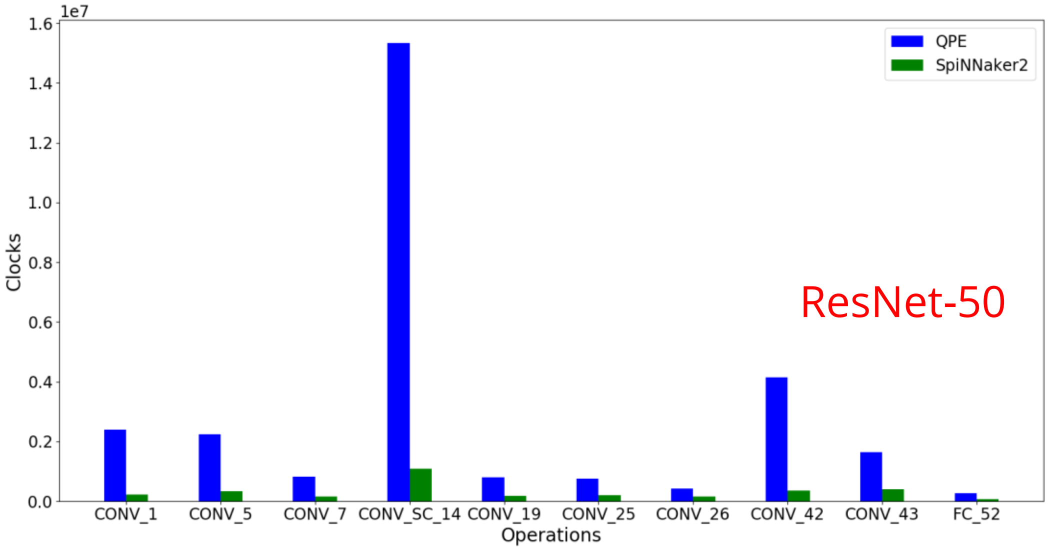
VGG-16:
Input : 502522 → 4.64%
weight: 468593 → 4.33%
Weight migration: 362723 → 3.35%

ResNet-50:
Input : 809929 → 16.62%
weight: 826177 → 16.95%
Weight migration: 109903 → 2.26%

Comparison of QPE and SpiNNaker2

Without operator fusion and without data reuse



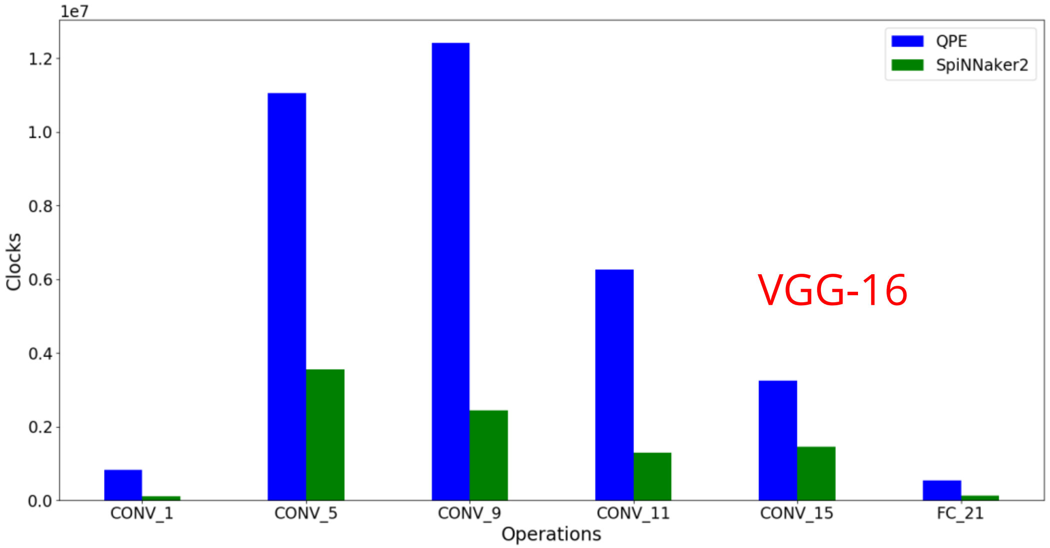


Up to 5 times

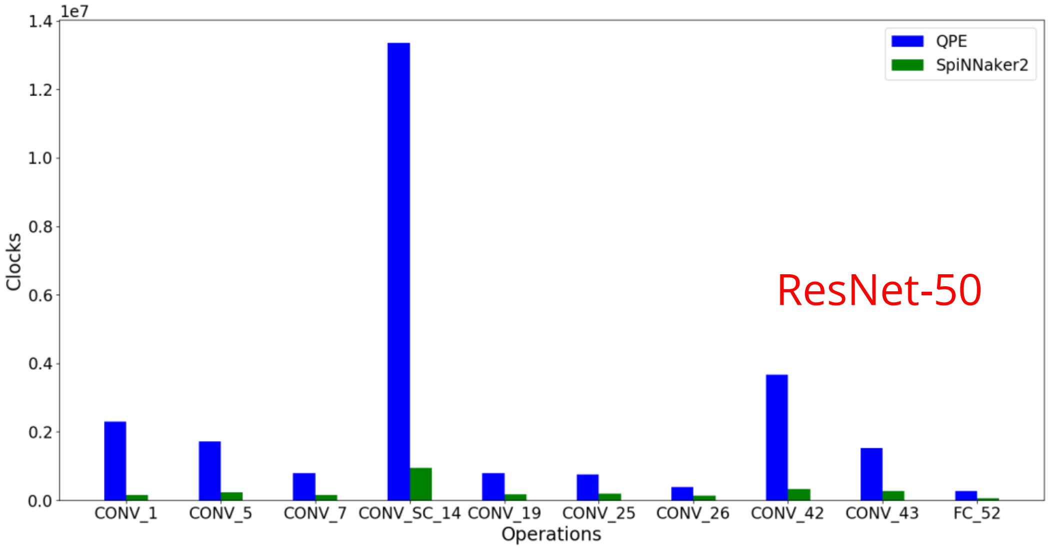
Up to 12 times

Comparison of QPE and SpiNNaker2

With operator fusion and without data reuse



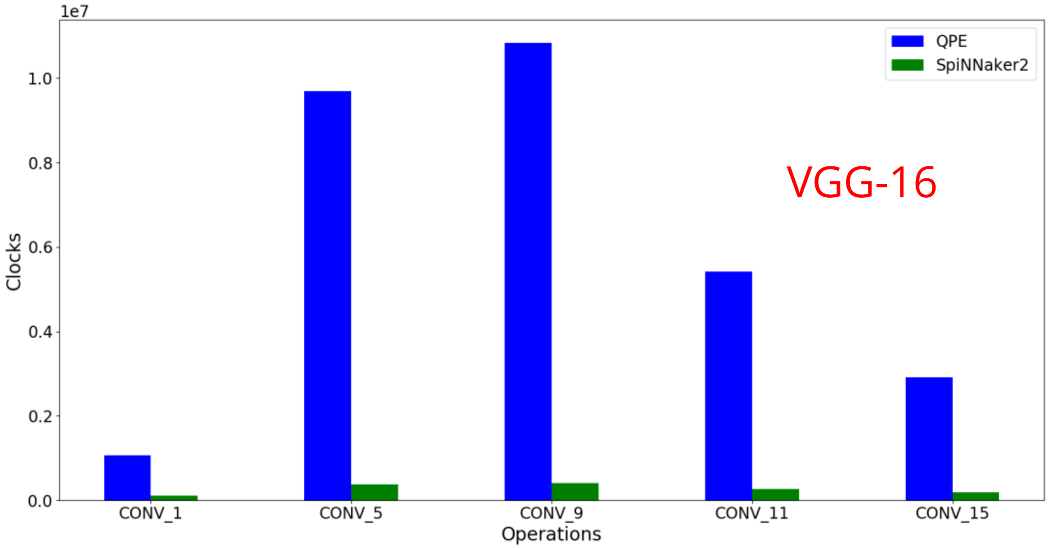
Up to 5 times

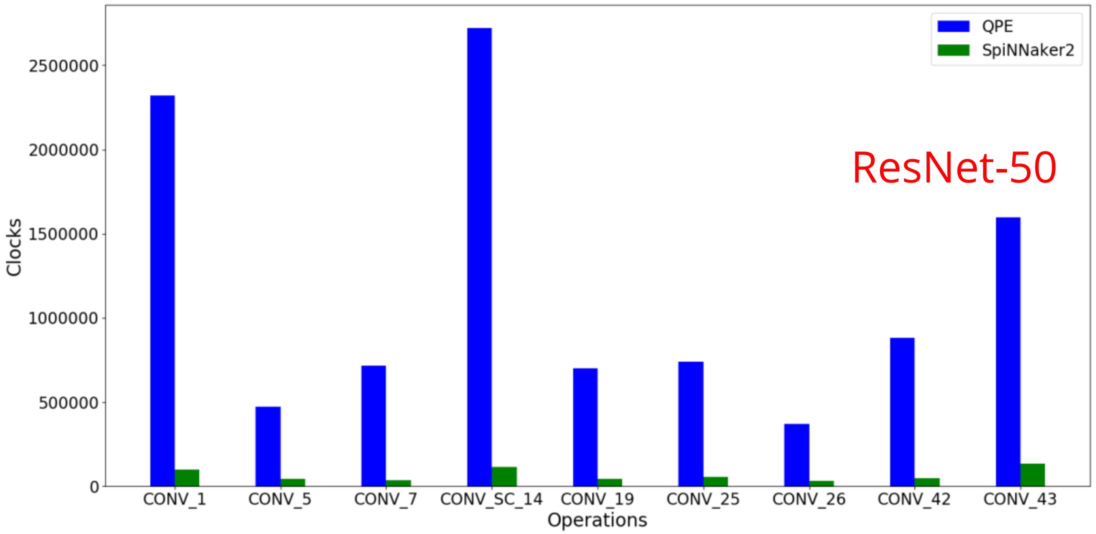


Up to 15 times

Comparison of QPE and SpiNNaker2

With operator fusion and With data reuse





Up to 26 times

Up to 24 times

Validation, Simulation and Experiment

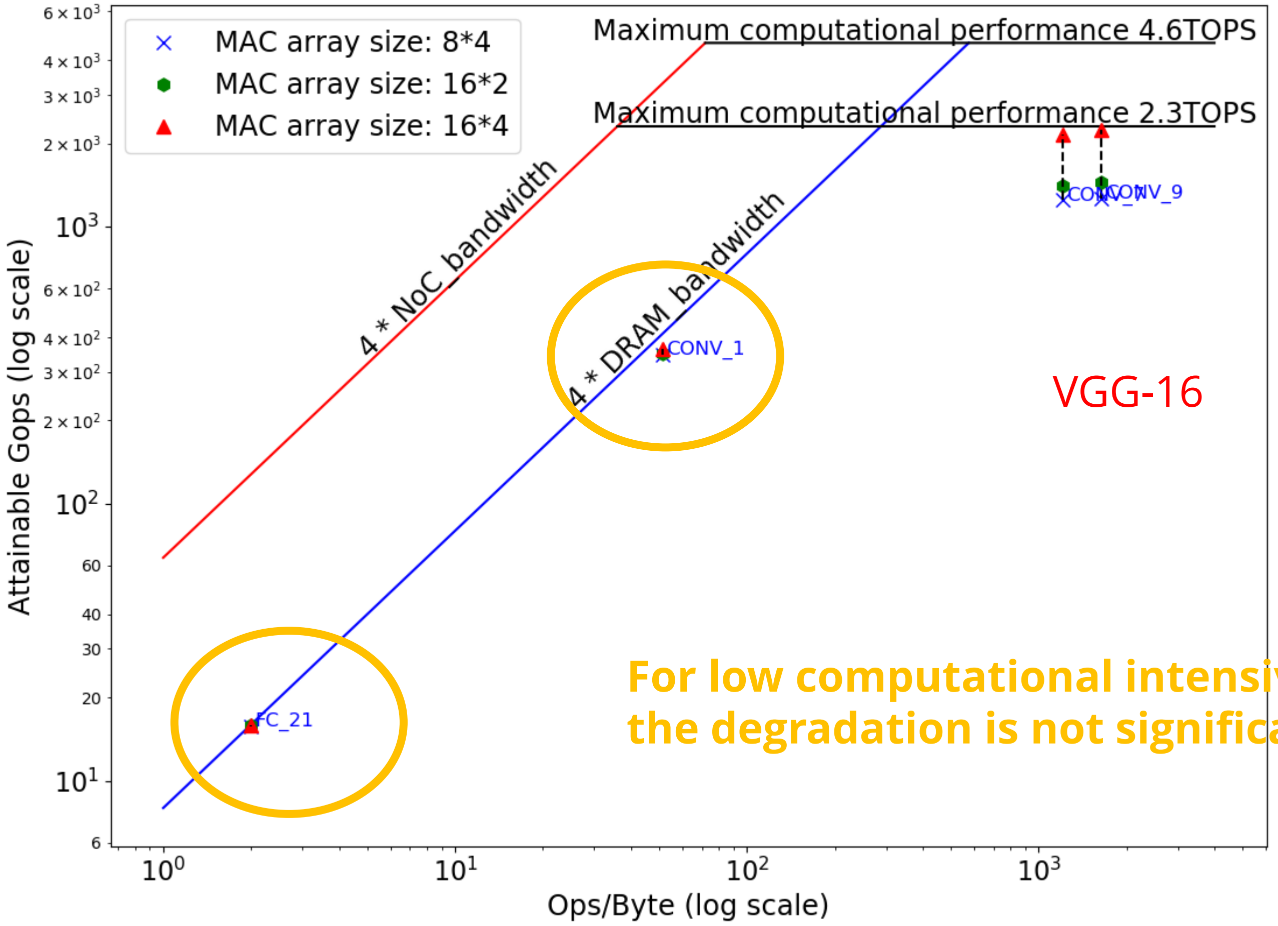
Experiment: Comparison between MLAs with different number of MAC units

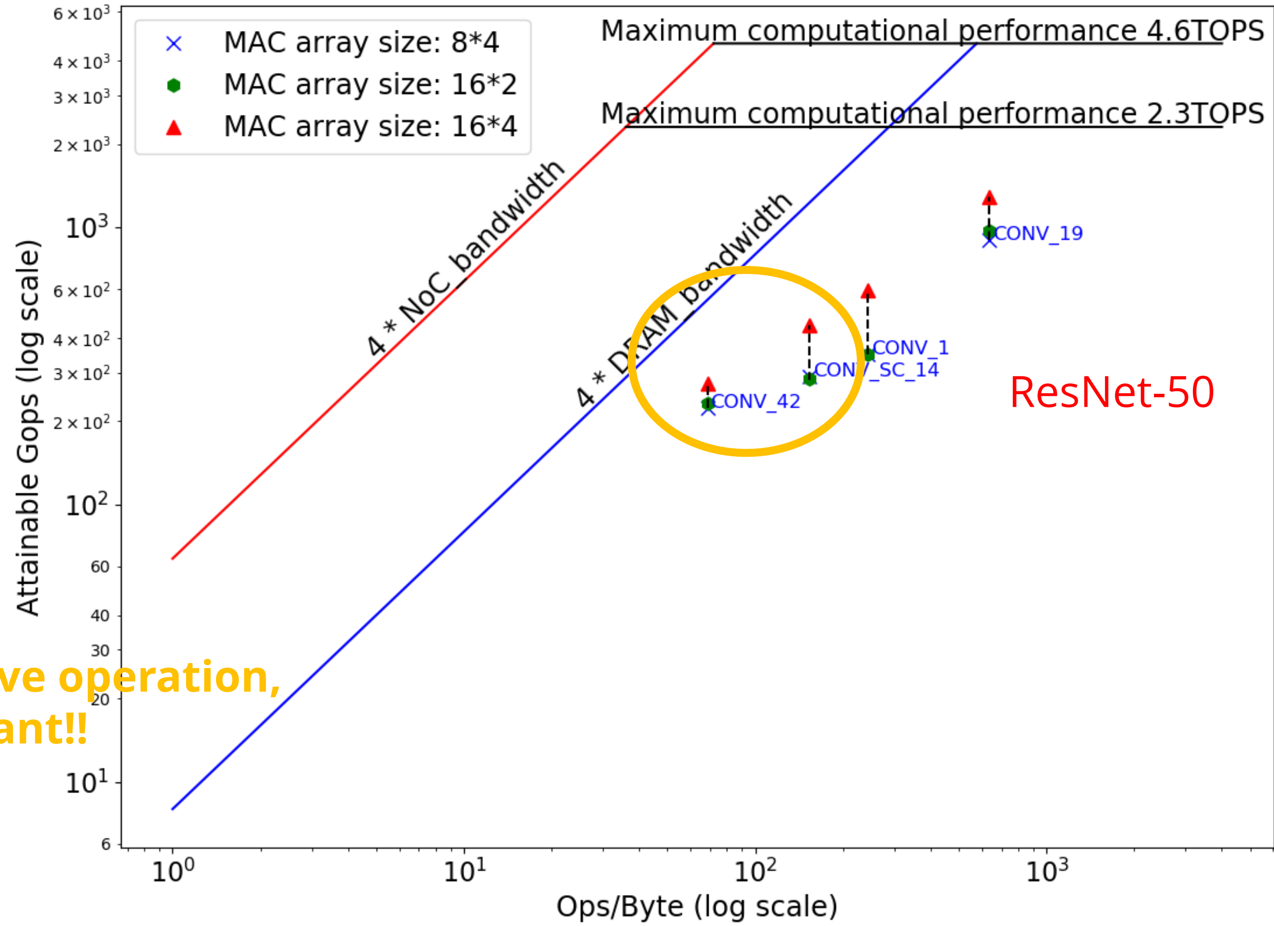
Operation	Operational intensity (operations/byte)	Performance [16*4] (Gops)	Performance [16*2] (Gops)	Performance [8*4] (Gops)
CONV_1	51.51	363.53 → 1.04 →	348.86	345.50
CONV_7	1210.28	2155.06 → 1.53 →	1410.20	1246.84
CONV_9	1641.38	2235.36 → 1.55 →	1445.07	1267.11
FC_21	2.00	15.88 → 1.00 →	15.90	15.79

Operation	Operational intensity (operations/byte)	Performance [16*4] (Gops)	Performance [16*2] (Gops)	Performance [8*4] (Gops)
CONV_1	243.44	592.30 → 1.69 →	350.29	350.01
CONV_SC_14	153.91	445.98 → 1.57 →	284.02	290.03
CONV_19	636.93	1284.88 → 1.32 →	976.13	898.47
CONV_42	68.50	272.63 → 1.17 →	232.21	223.96

Validation, Simulation and Experiment

Experiment: Comparison between MLAs with different number of MAC units





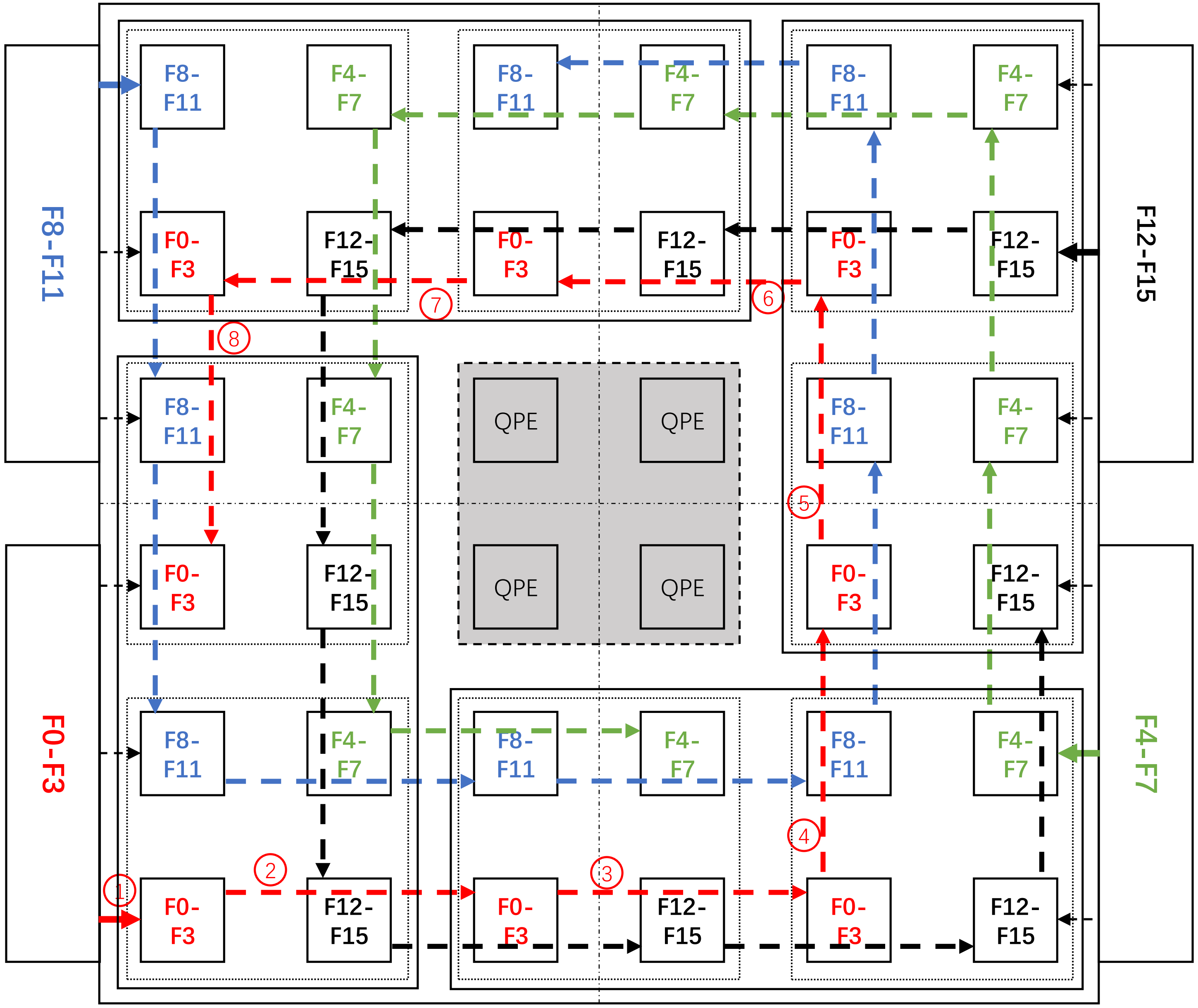
For low computational intensive operation, the degradation is not significant!!

Comparison the simulation result of QPE and SpiNNaker2

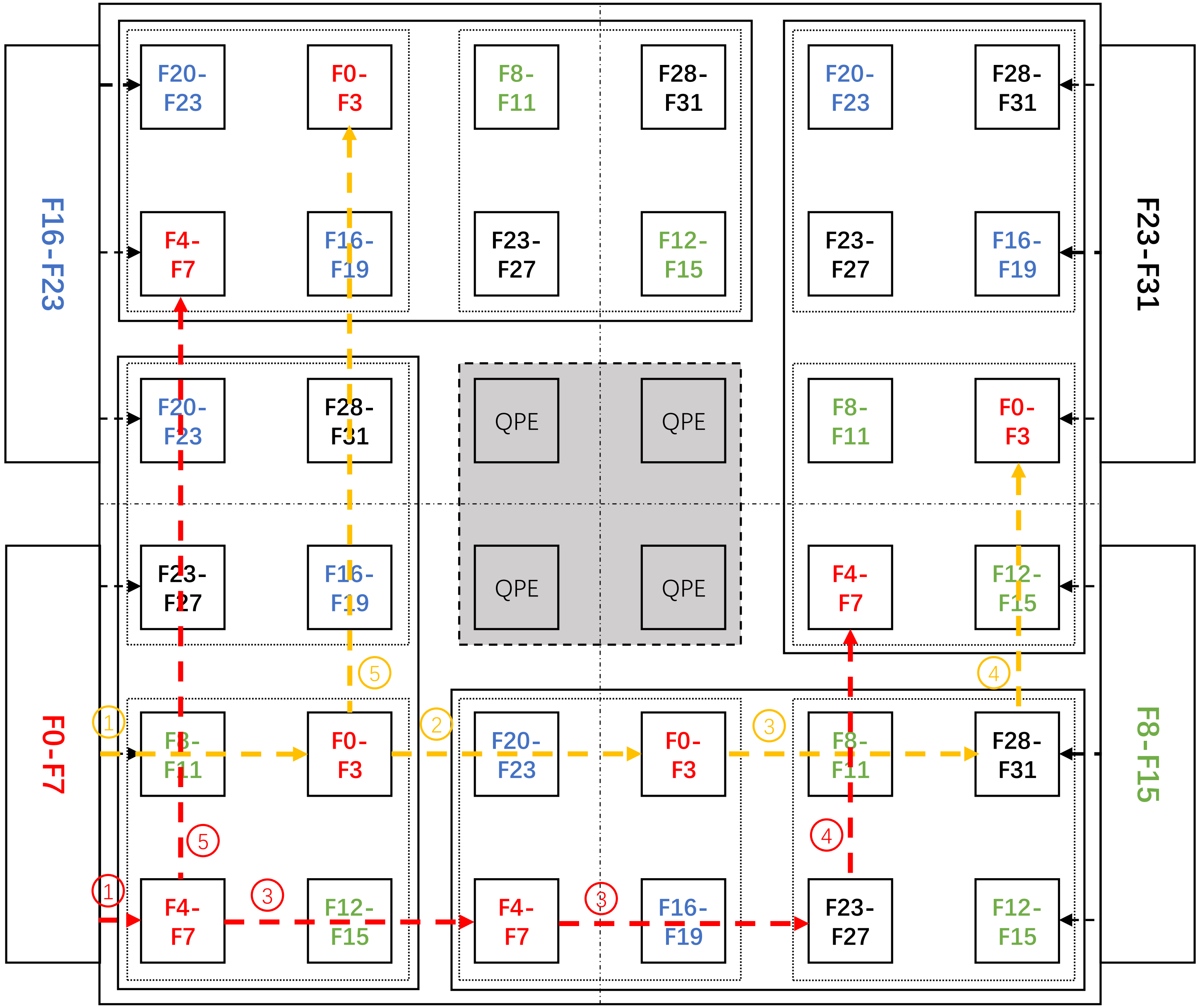
Distribution algorithm	Improvement of SpiNNake2 against QPE (VGG-16)	Improvement of SpiNNake2 against QPE (ResNet-50)
Without operator fusion and without data reuse	Up to 5 times	Up to 12 times
With operator fusion and without data reuse	Up to 5 times	Up to 15 times
With operator fusion and With data reuse	Up to 26 times	Up to 24 times

Improvement of SpiNNake2 against QPE should be in [4, 36]

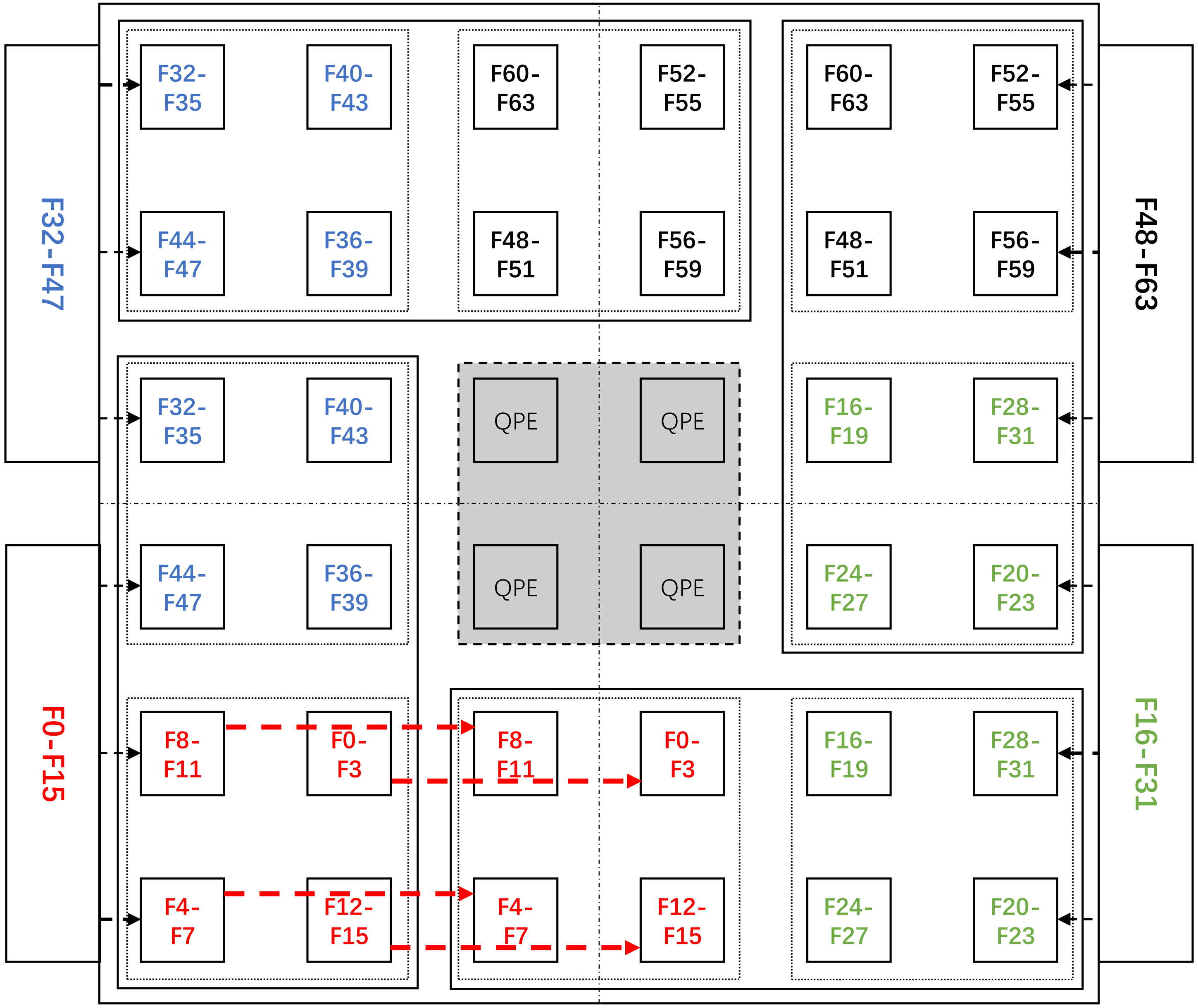
16 parts of filter weight



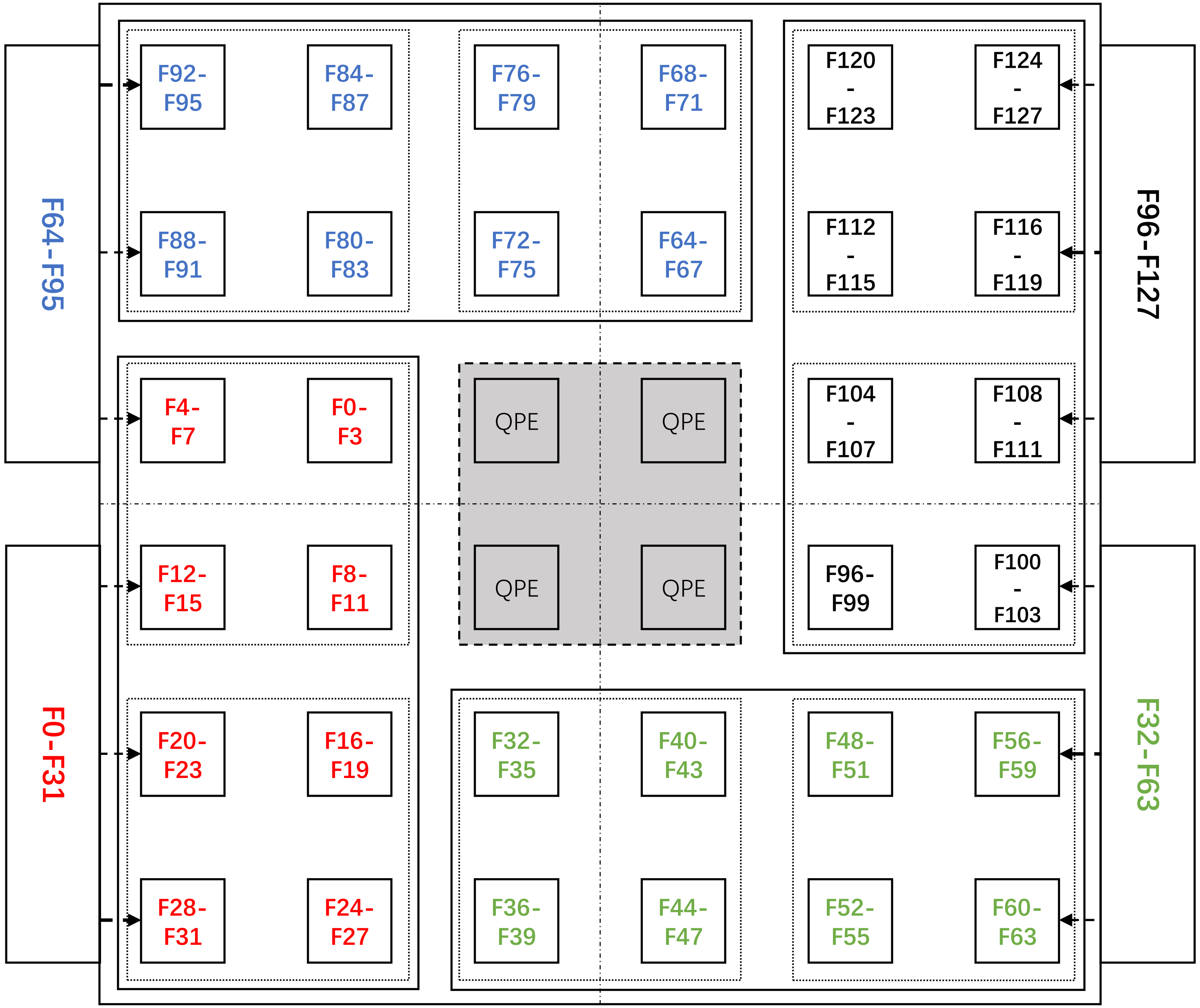
32 parts of filter weight

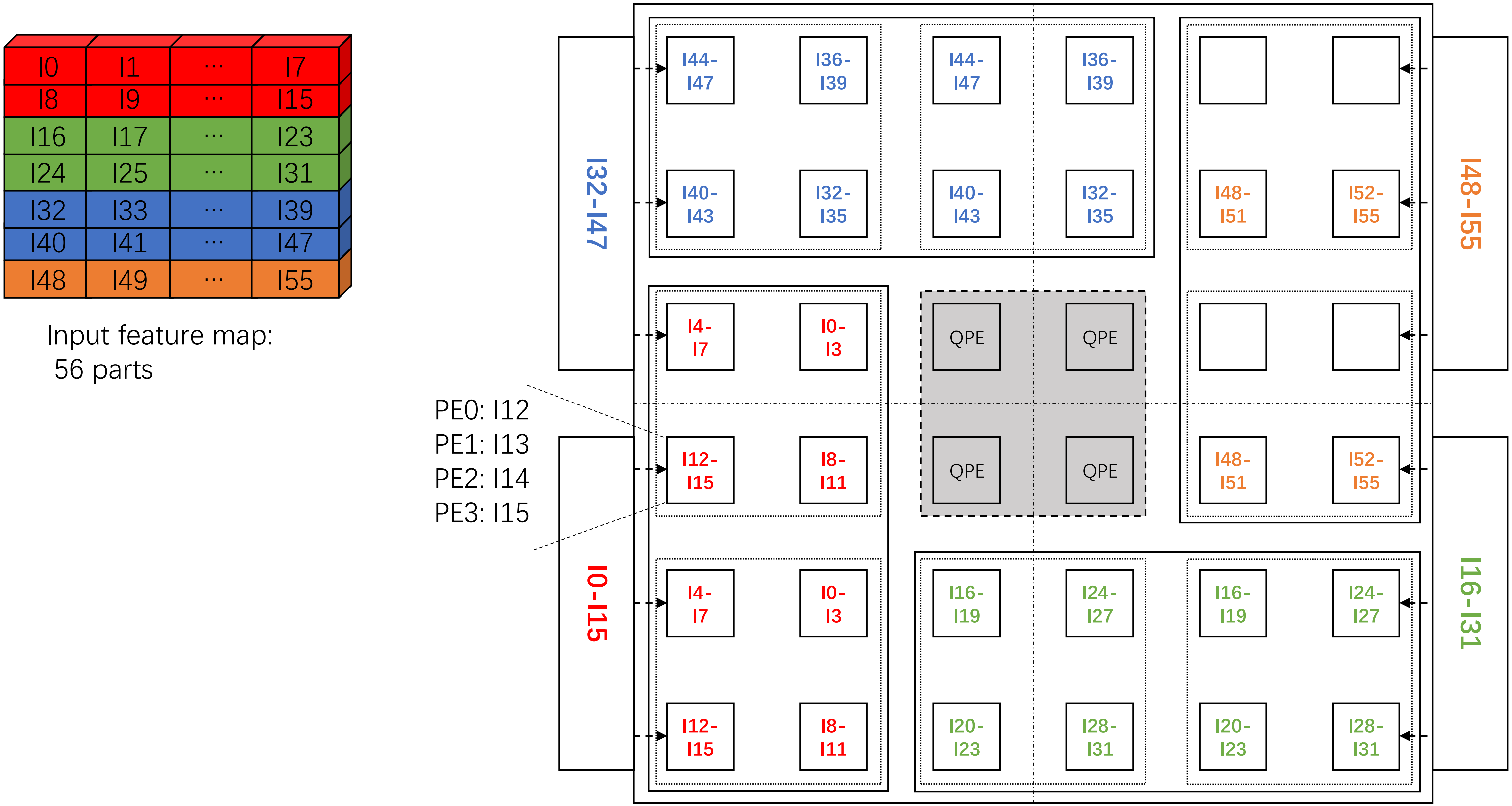


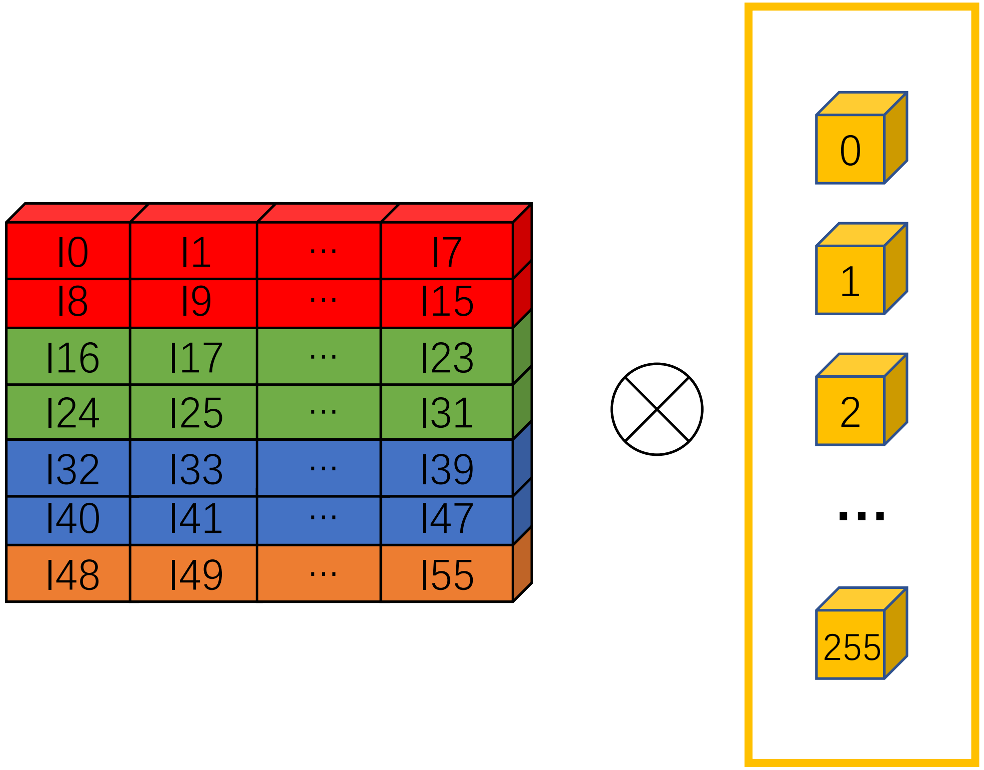
64 parts of filter weight



64 parts of filter weight




I0 I1 ··· I7
I8 I9 ··· I15
I16 I17 ··· I23
I24 I25 ··· I31
I32 I33 ··· I39
I40 I41 ··· I47
I48 I49 ··· I55
Input feature map:
56 parts
PE0: I12
PE1: I13
PE2: I14
PE3: I15
I32-I47
I0-I15
I48-I55
I16-I31
I44-I47
I36-I39
I40-I43
I32-I35
I4-I7
I0-I3
I12-I15
I8-I11
QPE
I48-I51
I52-I55
I16-I19
I24-I27
I20-I23
I28-I31



Split into 64 parts, each part has 4 filter

$$56 * 64 = 3584 \text{ tasks}$$

Step 1: After fetching input and weight into SpiNNaker2.

Using data reuse in QPE:

(28*4)*4 = 448 tasks

are finished.

Step 2: Migrate filter inside QPE block

Using data reuse in QPE:
(28*4)*4 = 448 tasks
are finished.

Step 3: Migrate filter inside QPE block

Using data reuse in QPE:
(28*4)*4 = 448 tasks
are finished.

Step 4: Migrate filter inside QPE block

Using data reuse in QPE:

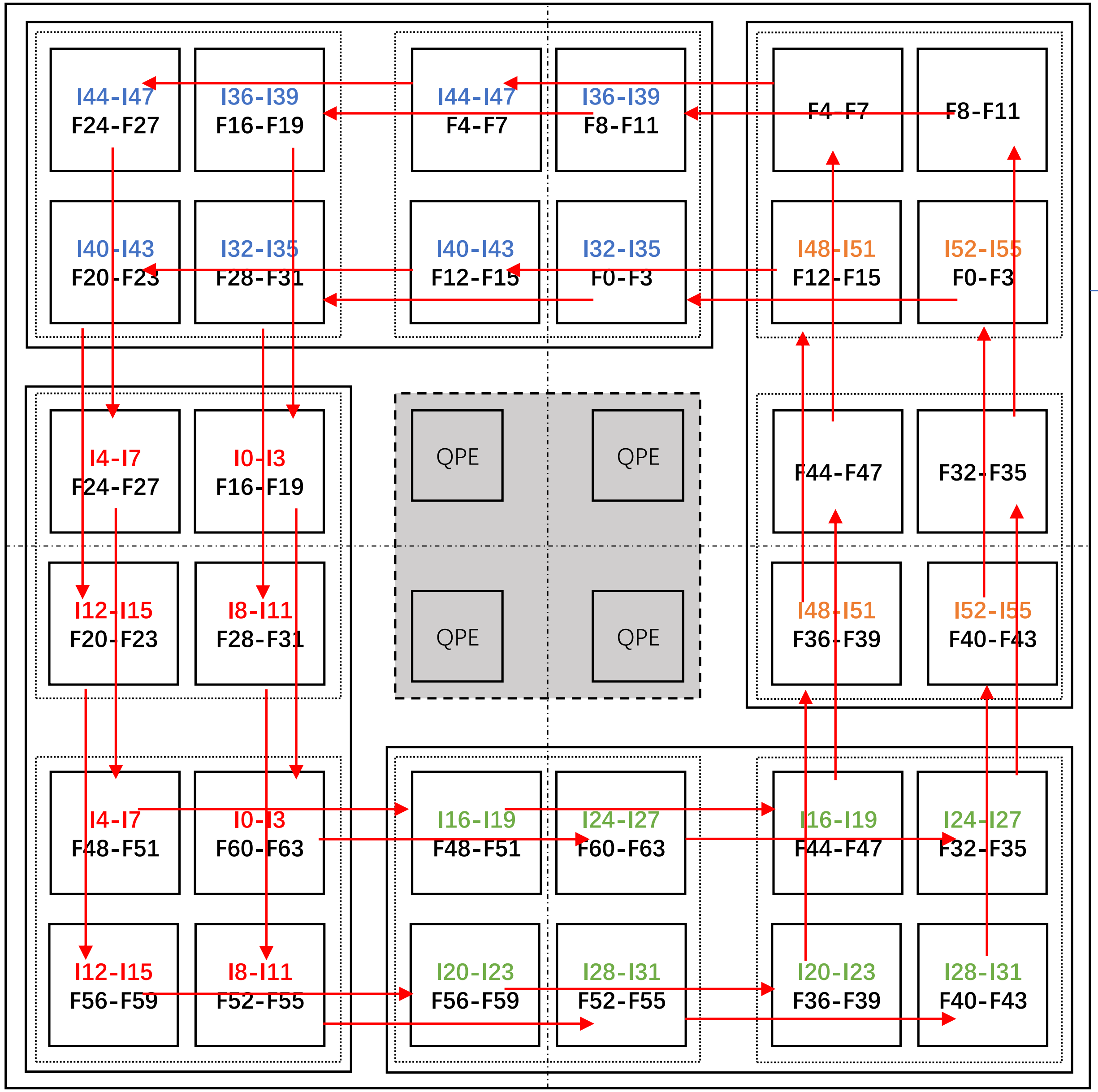
(28*4)*4 = 448 tasks are finished.

448 * 4 = 1792 tasks are finished.

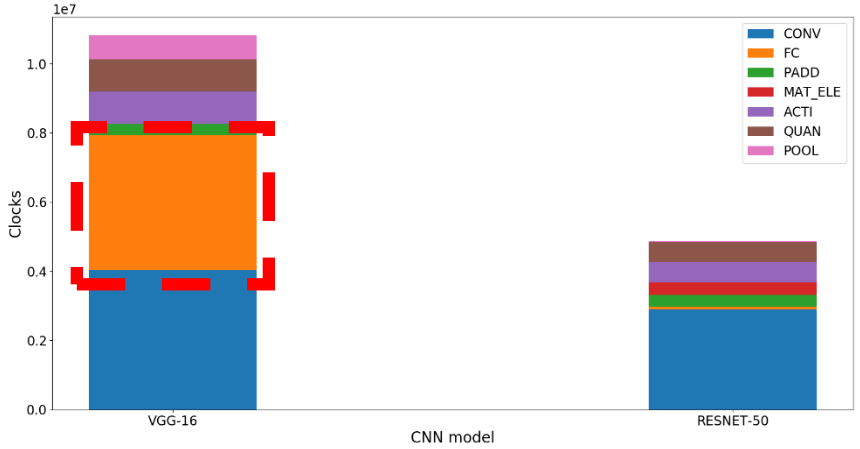
Step 5: Migrate filter inside SpiNNaker2

Repeat step 1-4:
448 * 4 = 1792 tasks are finished.

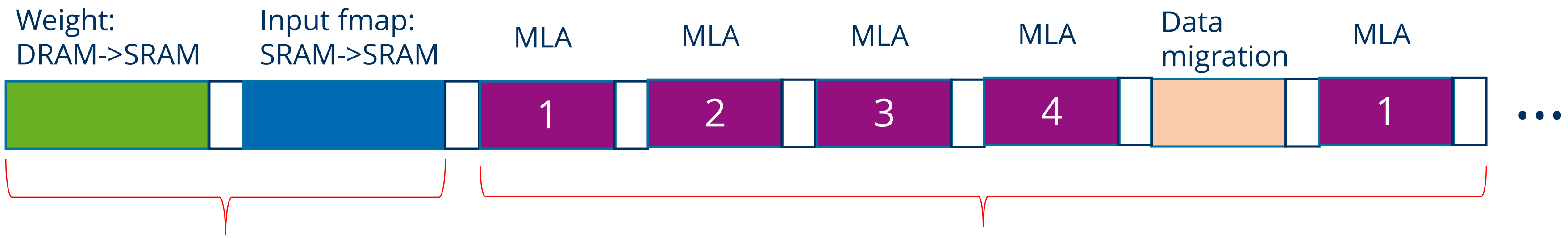
After then
All 3584 tasks are finished.



Conclusion: improvements on Distributor



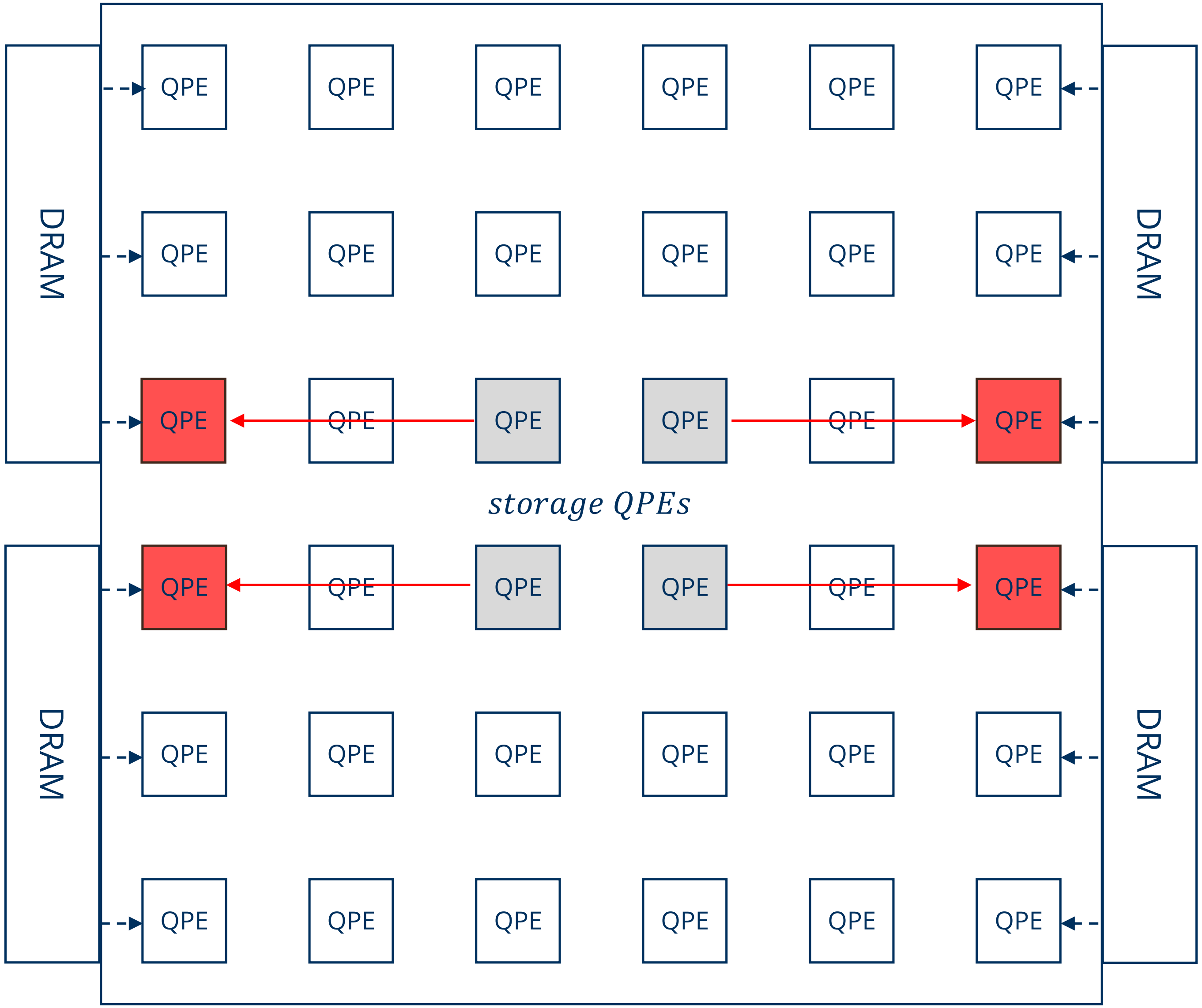
With operator fusion and with data reuse



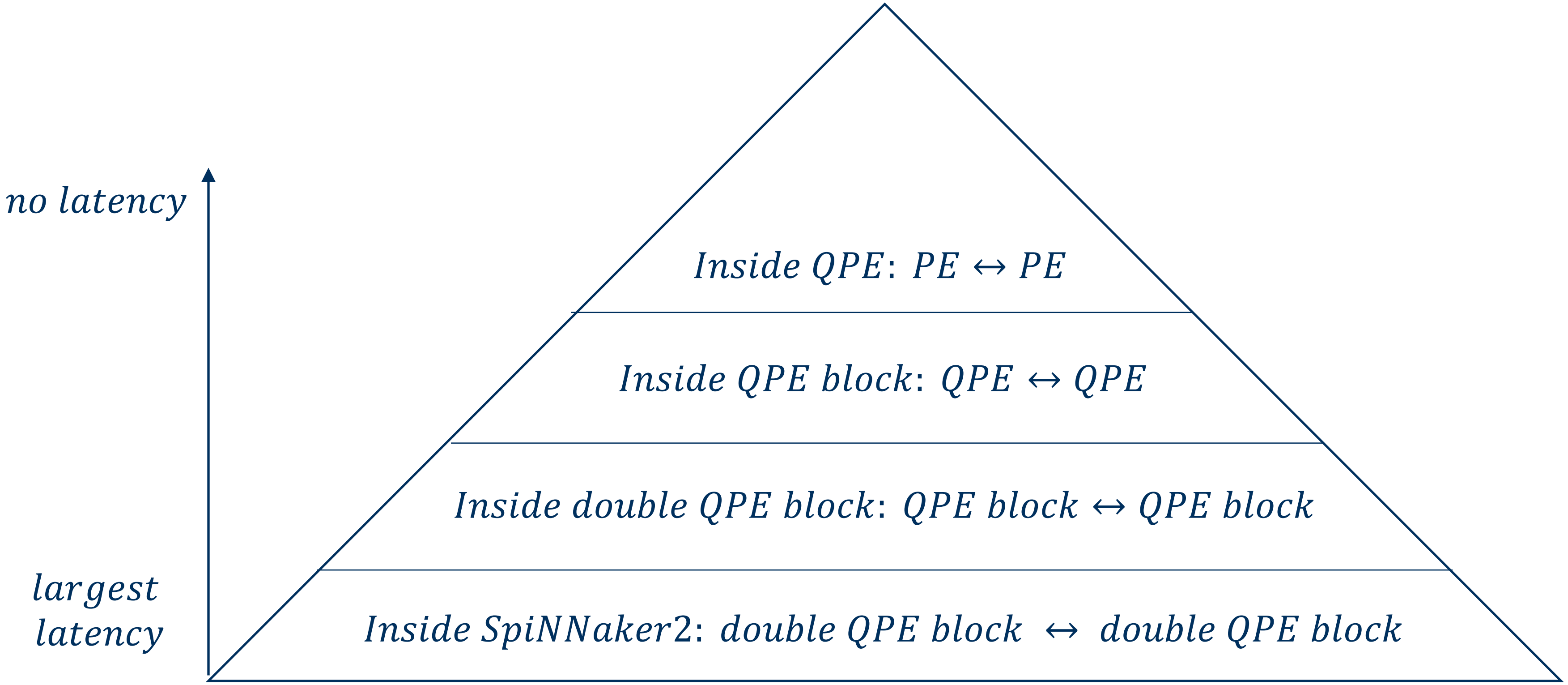
① Can be done at same time

②DRAM is idle during acceleration.
Pre-caching weight for next operation, especially for MM

Conclusion



xxx



no latency
largest latency
Inside QPE: PE ↔ PE
Inside QPE block: QPE ↔ QPE
Inside double QPE block: QPE block ↔ QPE block
Inside SpiNNaker2: double QPE block ↔ double QPE block