Florian Kelber*, Binyi Wu*, Bernhard Vogginger, Johannes Partzsch, Chen Liu, Marco Stolba, Christian Mayr
Faculty of Electrical and Computer Engineering
Chair of Highly-Parallel VLSI Systems and Neuro-Microelectronics

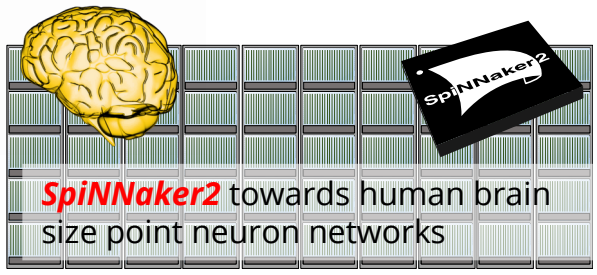
Mapping Deep Neural Networks on SpiNNaker2

NICE2021 16.03.2021

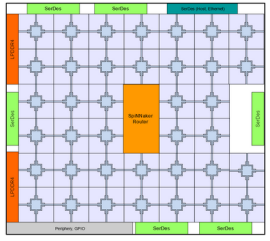
Table of Contents

Roadmap

2023
2022
2021
2020
2019
2018
2017
2016
2015
2014
2013

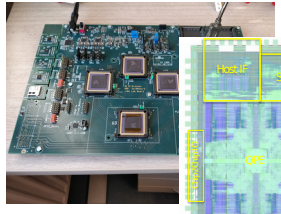


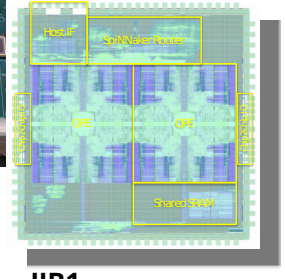
SpiNNaker2 towards human brain size point neuron networks

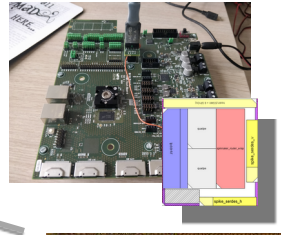


SpiNNaker2

- **152 Arm Cortex-M4F**
- power management
- SpiNNaker router
- low swing serial I/O
- 2x LPDDR4 memory IF
- 1GByte LPDDR4 PoP
- **22FDX** CMOS





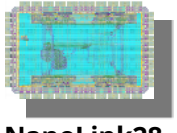


JIB2

- **8 Arm Cortex-M4F**
- SpiNNaker2-ready components
- LPDDR4 proof-of-concept

JIB1

- **8 Arm Cortex-M4F**
- SpiNNaker router,
- low swing serial I/O
- **22FDX** CMOS



NanoLink28

- SerDes Transceiver
- 28nm CMOS

Santos28

- **4 Arm Cortex-M4F**
- Power management
- SpiNNaker router with SerDes
- LPDDR2 Memory Interface
- 28nm CMOS

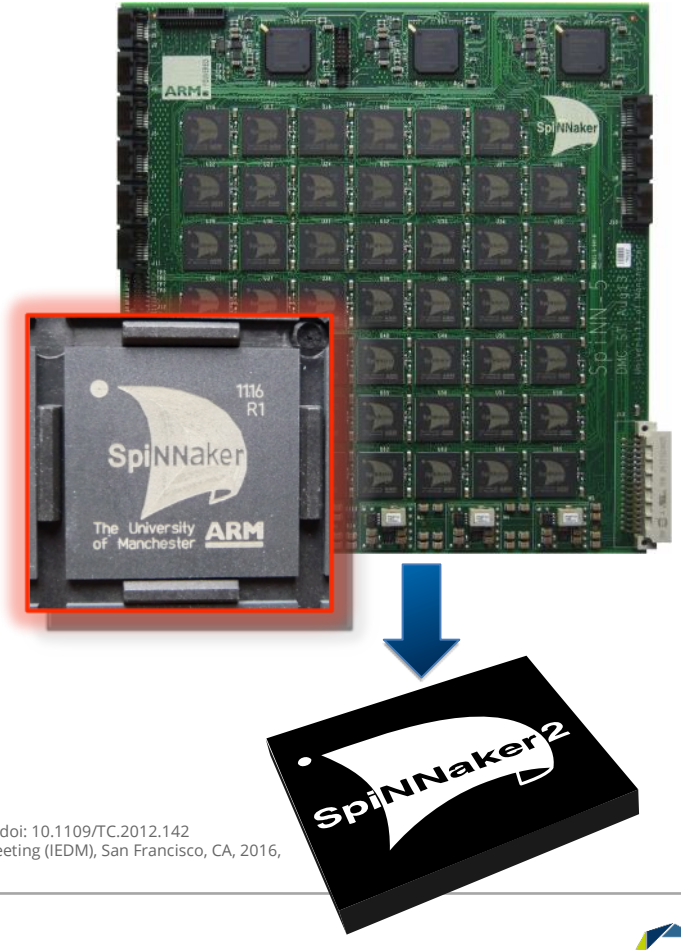
Human Brain Project

Towards SpiNNaker2

- Communication and memory centric architecture for efficient real-time simulation of spiking neural networks [1]

- Improvements:
  - 130nm CMOS -> 22FDX [2]
  - 18 -> **152 PEs** (ARM M4F cores) per chip
  - 200MHz -> 300MHz PE max clk
  - DRAM 128MB 1GB/s -> **1GB LPDDR4 2x3.2 GB/s**
  - single precision floating point unit
  - ECC SRAM, **NoC**
  - DVFS, PSO, ABB
  - EXP/LOG, PRNG/TRNG, **MATMUL/CONV2D** accelerators

- SpiNNaker2 target: Enhance capacity for brain size network simulation in real time at >10x better efficiency



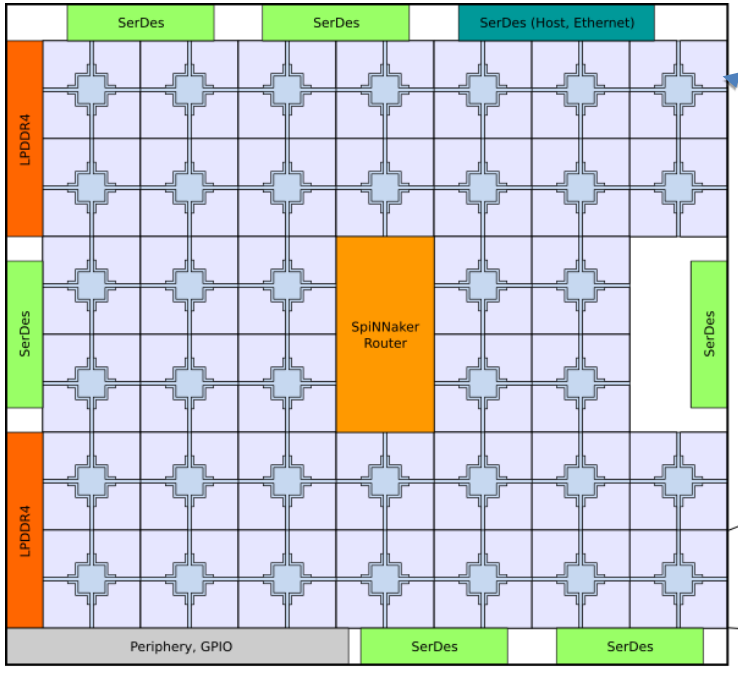
[1] S. B. Furber et al., "Overview of the SpiNNaker System Architecture," in IEEE Transactions on Computers, vol. 62, no. 12, pp. 2454-2467, Dec. 2013. doi: 10.1109/TC.2012.142
[2] R. Carter et al., "22nm FDSOI technology for emerging mobile, Internet-of-Things, and RF applications," 2016 IEEE International Electron Devices Meeting (IEDM), San Francisco, CA, 2016, pp. 2.2.1-2.2.4. doi: 10.1109/IEDM.2016.7838029

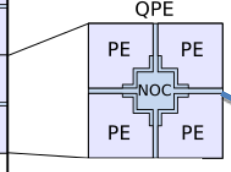
Architecture for Efficient DNN Inference

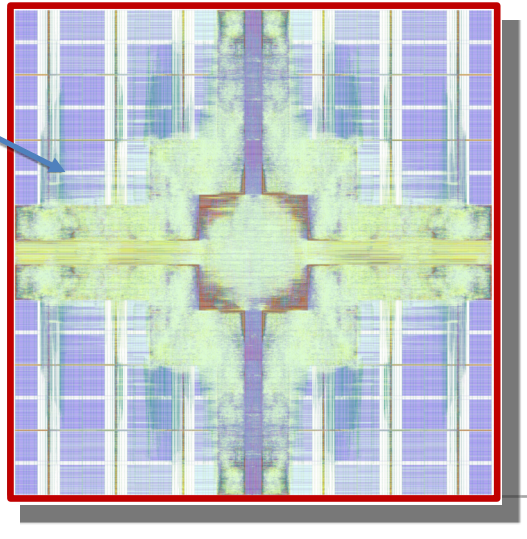
Motivation: Efficient DNN Inference on SpiNNaker2

1. Make SpiNNaker2 DNN compatible -> Support Hybrid Nets
2. Support general higher scale conv, fc layers exploiting SpiNNaker2 distributed structure with minimal area/power increase
3. Maximize fps and TOPS/W
    1. Add MAC array accelerator
    2. Minimize off-chip DRAM Access
    3. Maximize utilization of data lanes/PEs
    4. Exploit GALS architecture for Decoupled Access/Execute
4. (Exploit fine-grained power management)

Architecture Overview

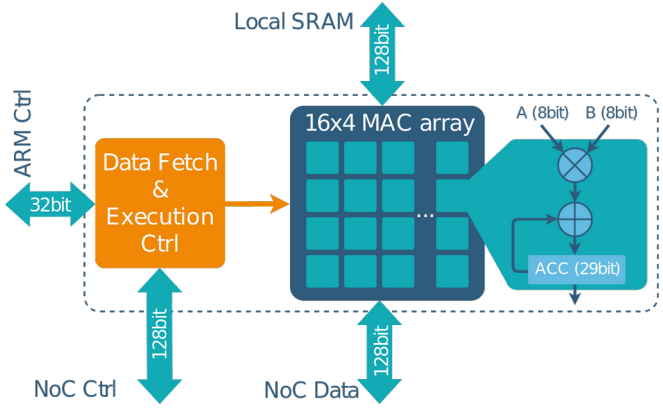


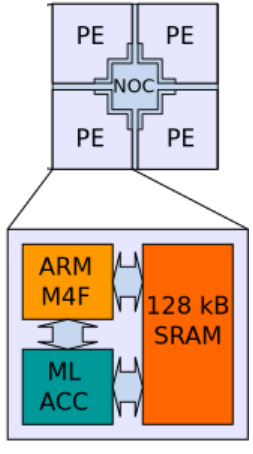




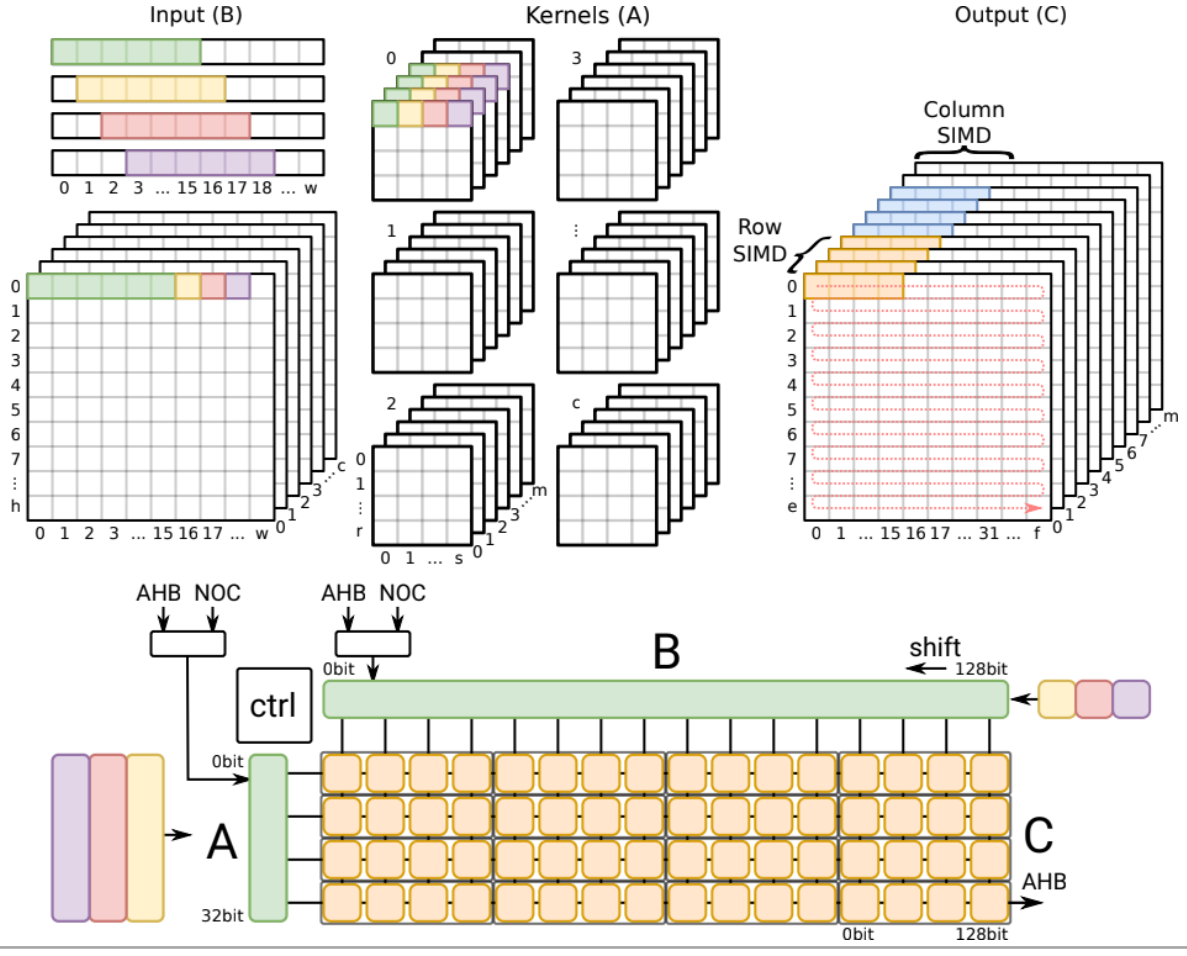
The Machine Learning Accelerator for SpiNNaker2

- 8b/16b signed/unsigned SIMD execution of matrix multiplication (MatMul) and 2d convolution (CONV)
- Functional parallelism due to multiprocessor architecture → each PE has it's own accelerator
- Decoupled access/execute for NoC remote source, zero gating, stride writeout, adder tree
- PE-level data reuse of output feature map → no input/output buffer needed but trade-off with weight data reuse





Conv2D example on the Accelerator



Mapping DNNs

Problem statement:

Common Network Models:

- Current SOTA neural computing is memory hungry (e.g. ResNet still backbone for many networks)
- Single PE has 128kByte memory -> e.g. worst ResNet-50 layer ≈ 12.41 MB -> 100 PEs needed
- Costly data fetching from DRAM + highly-distributed nature of SpiNNaker2

Solution:

- Automatic shared fork-join nested parallelism [3] -> divide tasks
- Deterministic, model/hardware-dependent row stationary [4] mapping of all subtasks to hardware
- Ensure high-% input data reuse during divide to minimize data movement
- Optimizing Computational Graph with Operator Fusion

[3] 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
[4] V. Sze, Y. Chen, T. Yang, and J. S. Emer. "Efficient Processing of Deep Neural Networks: A Tutorial and Survey". In: Proceedings of the IEEE 105.12 (2017), pp. 2295–2329. issn: 0018-9219. doi: 10.1109/ JPROC.2017.2761740

Efficient CNN Mapping on SpiNNaker2:

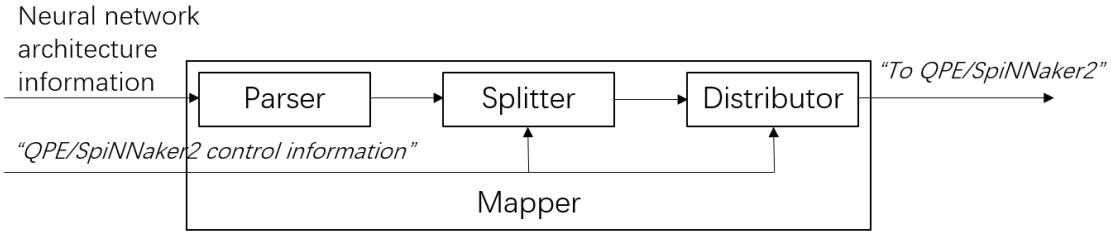
Objective:

- Find mapping and scheduling algorithms optimizing for **lowest latency** of **CNN** inference on SpiNNaker2

Approach:

- **Python Simulator** that emulates relevant components and replicate timing of CNN execution and data transfer / congestions

- **Parser**: Operator Fusion
- **Splitter**: Nested Task Parallelism
- **Distributor**: Mapping and Scheduling



$$convolutional\ layer \rightarrow \begin{cases} padding\ operation\ (ARM) \\ convolution\ operation\ (MLA) \\ nonlinearity\ operation\ (ARM) \\ quantization\ operation\ (ARM) \end{cases}$$

$$pooling\ layer \rightarrow \begin{cases} padding\ operation\ (ARM) \\ pooling\ operation\ (MLA/ARM) \end{cases}$$

$$fully-connected\ layer \rightarrow \begin{cases} matrix\ multiplication\ operation\ (MLA) \\ nonlinearity\ operation\ (ARM) \\ quantization\ operation\ (ARM) \end{cases}$$

Efficient CNN Mapping on SpiNNaker2:

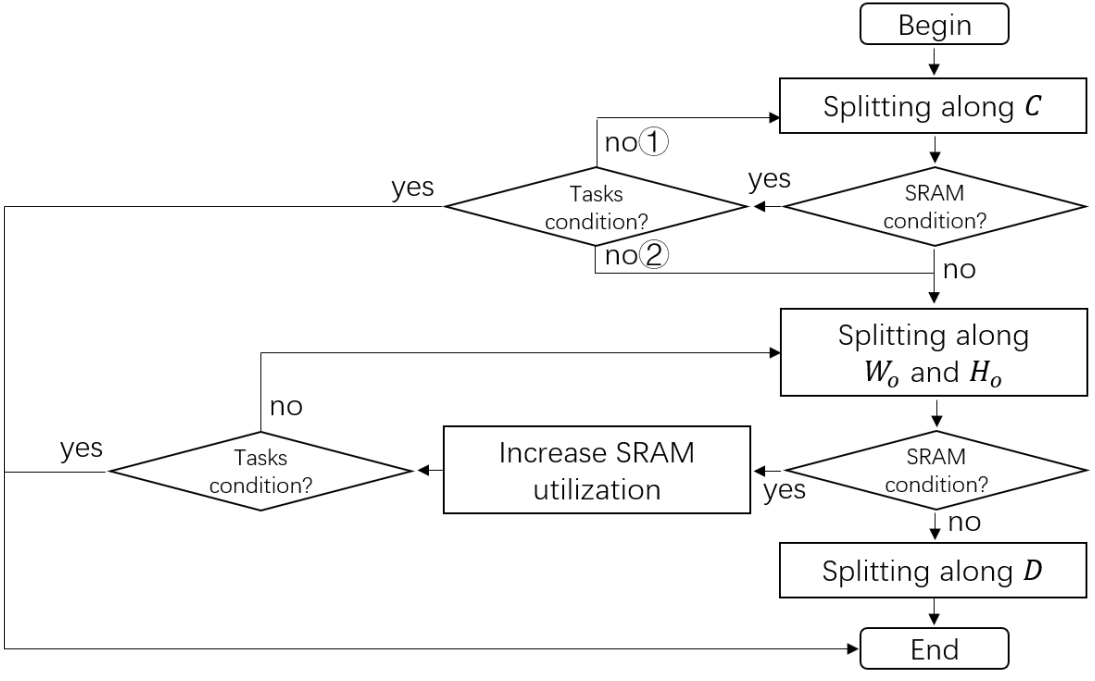
Objective:

- Find mapping and scheduling algorithms optimizing for **lowest latency** of **CNN** inference on SpiNNaker2

Approach:

- **Python Simulator** that emulates relevant components and replicate timing of CNN execution and data transfer / congestions

- **Parser**: Operator Fusion
- **Splitter**: Nested Task Parallelism
- **Distributor**: Mapping and Scheduling



SRAM condition: If the available SRAM is enough for storing the input feature map, weight and output feature map?
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 the O_C into more parts.
no②: The number of tasks cannot be increased by splitting the O_C into more parts.

Efficient CNN Mapping on SpiNNaker2:

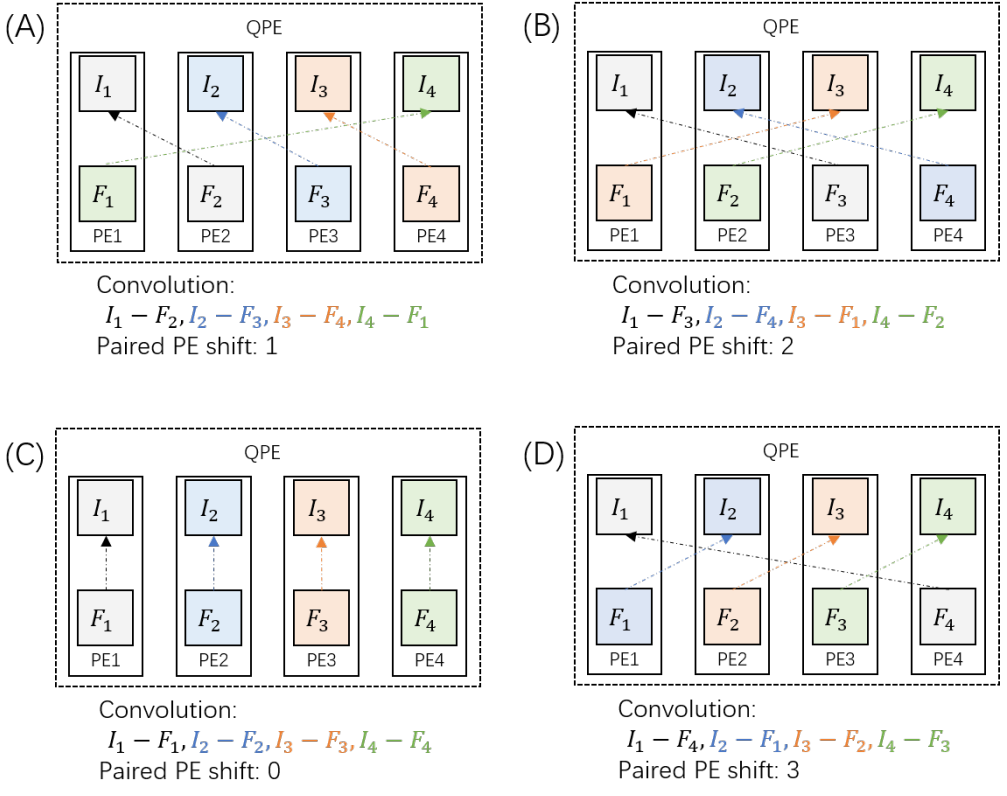
Objective:

- Find mapping and scheduling algorithms optimizing for **lowest latency** of **CNN** inference on SpiNNaker2

Approach:

- **Python Simulator** that emulates relevant components and replicate timing of CNN execution and data transfer / congestions

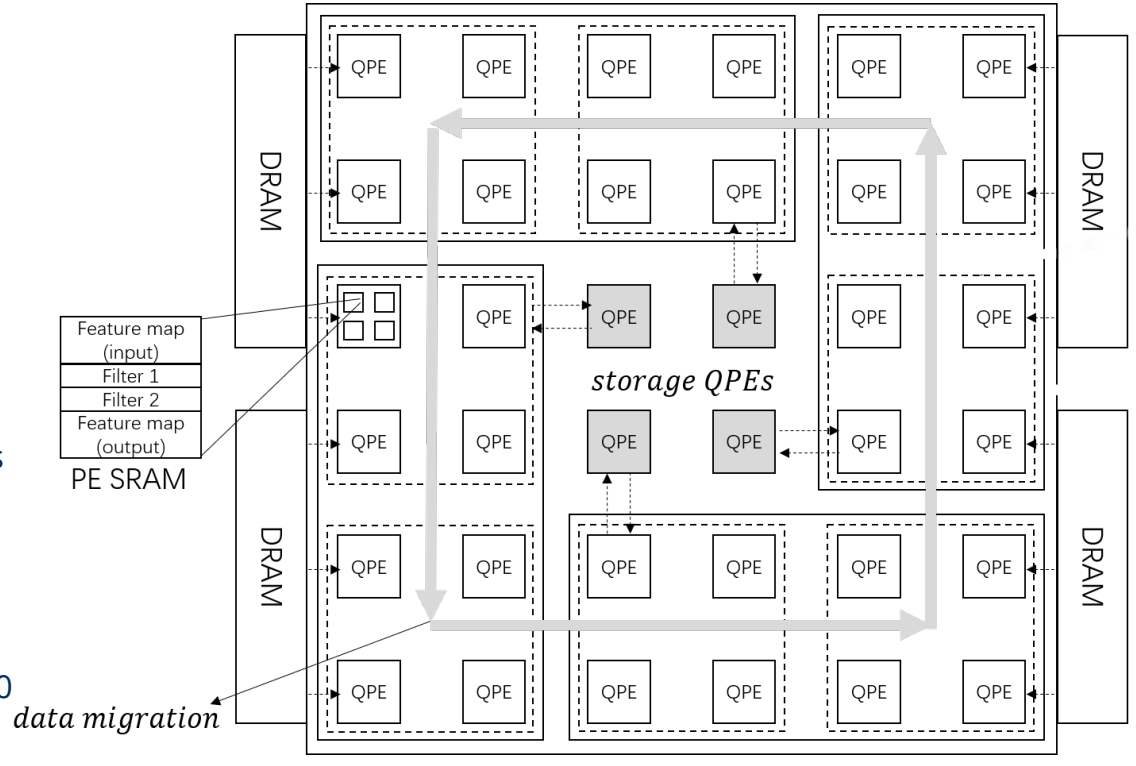
- **Parser**: Operator Fusion
- **Splitter**: Nested Task Parallelism
- **Distributor**: Mapping and Scheduling



Distribution & Scheduling:

Mapping Strategy:

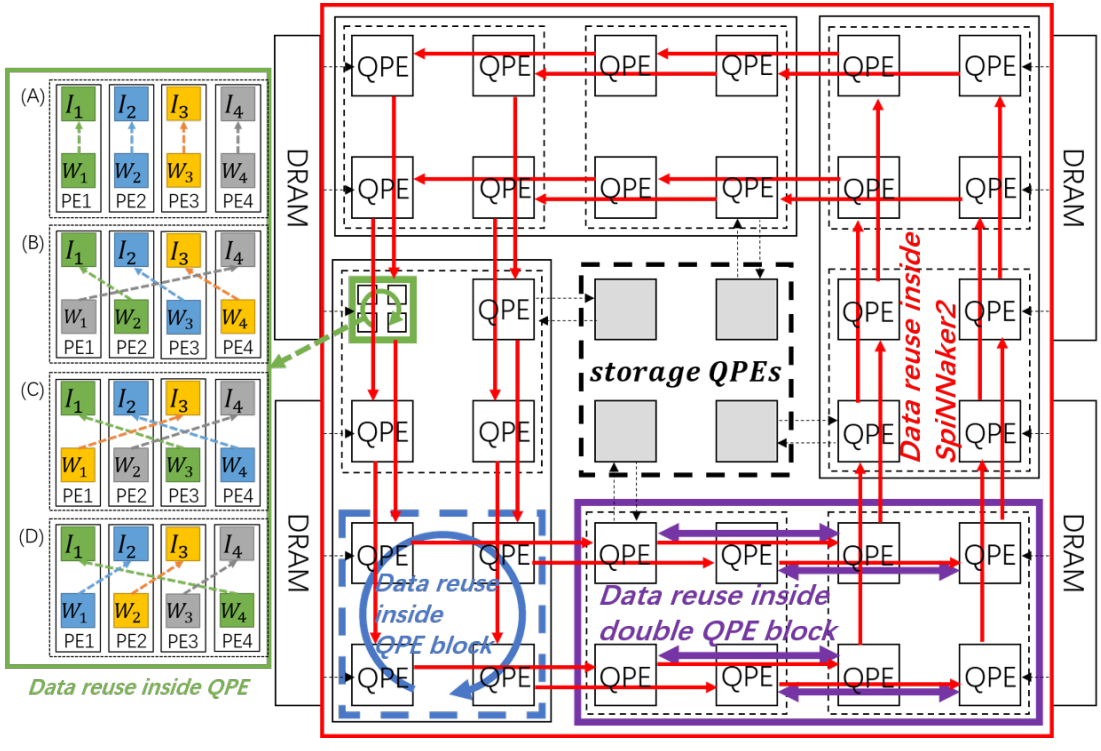
- Break PEs into **hierachies:**
  - QuadPE (QPE) = 4x PEs
  - QPE Block (QB) = 4x QPE
  - Double QPE Block (DQB) = 2x QPE block
- Different **mapping strategies**, scheduling, minimizing DRAM access through **data reuse**
- **Data migration** of the weights for many Infmap-Kernel combinations
- Evaluation for VGG-16 and ResNet-50



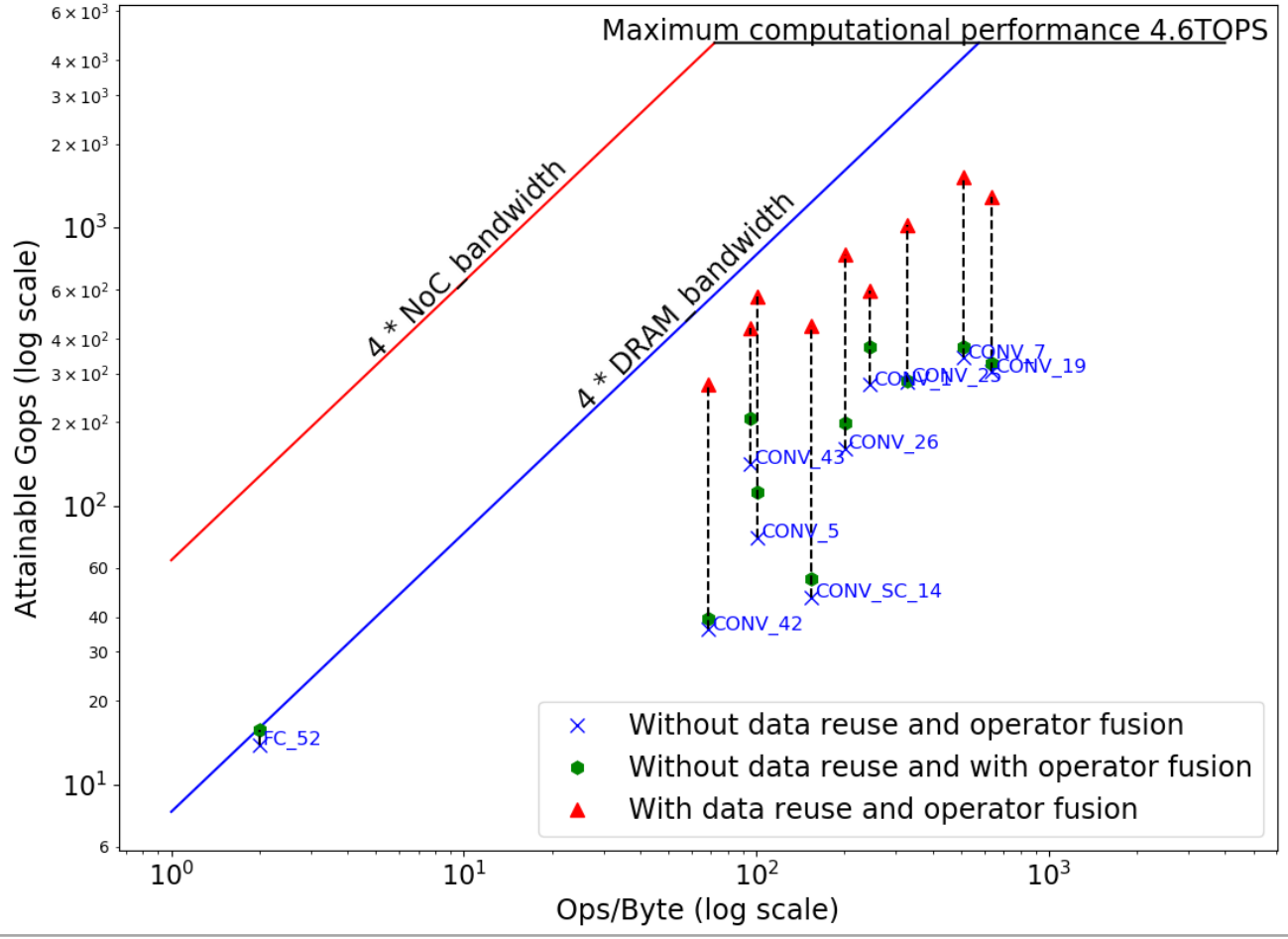
Distribution & Scheduling:

Mapping Strategy:

- Break PEs into **hierachies:**
  - QuadPE (QPE) = 4x PEs
  - QPE Block (QB) = 4x QPE
  - Double QPE Block (DQB) = 2x QPE block
- Different **mapping strategies**, scheduling, minimizing DRAM access through **data reuse**
- **Data migration** of the weights for many Infmap-Kernel combinations
- Evaluation for VGG-16 and ResNet-50

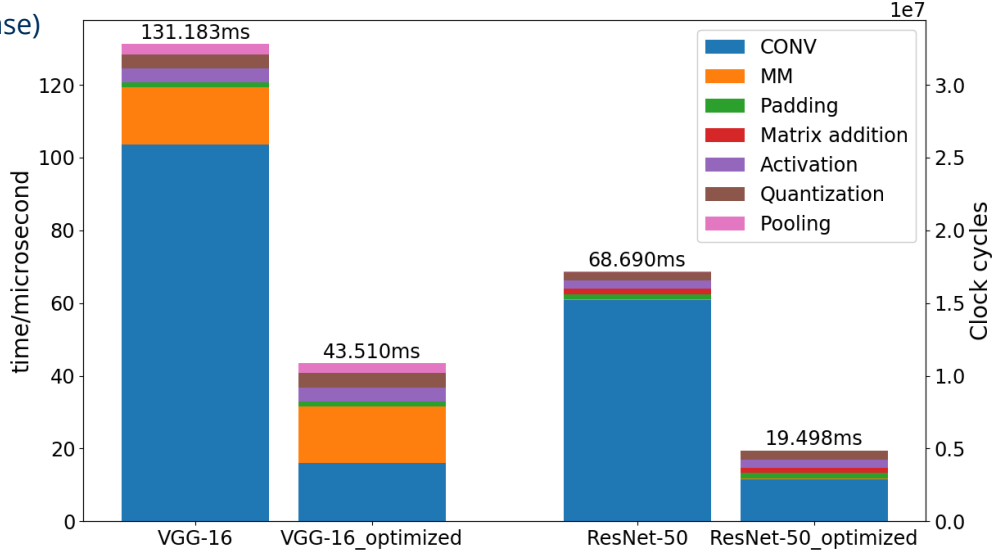


Roofline Model (ResNet-50)



Key Results

- Added broadcast output-stationary accelerator per pe with an area increase of 7% 22DFX -> 4.6TOPS @ 3.8 TOPS/W (Jib1 prototype measurement)
- SpiNNaker2Py to rapidly estimate data movement and ressources
- Through data reuse strategy increase inference fps by:
  - VGG-16: 7.6 fps -> 24.1 fps (x3.17 increase)
  - ResNet-50: 14.6 fps -> 51.3 fps (x3.51 increase)
- Future plans:
  - TapeOut SpiNNaker2
  - Energy-aware mapping/scheduling
  - Test out further models/applications



Thank you very much for your attention !