Mapping Deep Neural Networks on SpiNNaker2

Florian Kelber, Binyi Wu, Bernhard Vogginger, Johannes Partzsch, Chen Liu, Marco Stolba and Christian Mayr

April 7, 2020

Mapping Deep Neural Networks on SpiNNaker2

Florian Kelber*
Technische Universität Dresden
Florian.Kelber@tu-dresden.de

Binyi Wu*
Technische Universität Dresden
Binyi.Wu@tu-dresden.de
Infineon Technologies Dresden GmbH & Co. KG
Binyi.Wu@infineon.com

Bernhard Vogginger
Johannes Partzsch
Chen Liu
Marco Stolba
Christian Mayr
Technische Universität Dresden

ABSTRACT

SpiNNaker is an efficient many-core architecture for the real-time simulation of spiking neural networks. To also speed up deep neural networks (DNNs), the 2nd generation SpiNNaker2 will contain dedicated DNN accelerators in each processing element. When realizing large CNNs on SpiNNaker2, layers have to be split, mapped and scheduled onto 144 processing elements. We describe the underlying mapping procedure with optimized data reuse to achieve inference of VGG-16 and ResNet-50 models in tens of milliseconds.

KEYWORDS

Neural algorithms and machine learning, SpiNNaker2, deep neural networks, neuromorphic hardware

1 INTRODUCTION

Neuromorphic hardware provides efficient realizations of spiking neural networks (SNNs) and promises unprecedented energy efficiency for artificial intelligence [1, 2, 11]. Yet, SNNs lack the accuracy and robustness of commonly applied deep neural networks [8]. To overcome this gap, the 2nd generation SpiNNaker system integrates hardware accelerators for DNN layers in the individual processing elements (PEs). This flexibility allows to choose between spiking and deep neural networks depending on whatever is more efficient for the task at hand. This paper presents a systematic distribution strategy for large convolutional neural networks (CNNs) on SpiNNaker2, achieving inference times and energy efficiency that are orders of magnitude better than DNN applications on previous SpiNNaker systems [5, 9, 12] and competitive with dedicated DNN chips [7].

2 SPINNAKER2 ARCHITECTURE

The Spinnaker2 system is a many-core architecture conceptualized for the acceleration of neuromorphic processes [4, 6]. It consists of 36 Quad Processing Elements (QPEs), each containing 4 PEs. QPEs are arranged on a 2D grid and connected over a Network-on-Chip (NoC) with 4 neighbouring modules (see figure 1). Each NoC lane provides a maximum throughput of 16Byte per clock cycle. Data can be provided over several interfaces including 4 LPDDR4 DRAM connections. Each PE module includes a general purpose ARM-Cortex-M4F core, 128kByte SRAM for data and instruction memory, and specialized accelerators adapted for specific operations [4]. For the context of DNN layers we included a broadcast output-stationary 16x4 2D MAC array. It supports SIMD execution of both 2D convolution (CONV) and matrix multiplication (MM) operations. The accelerator can be started directly from the ARM core within the PE or from the global NoC. It independently fetches the input activations from the local SRAM and the weights via the NoC from a different PE, computes the CONV or MM with 64 MAC operations per clock cycle, and writes back the results to the local SRAM. The memory bandwidth amounts to 16 Bytes per clock cycle from the NoC and 16 Bytes read and write to the SRAM. Furthermore it reuses input feature map values by shifting them, therefore decreasing the feature map fetch from 16byte/clk to 4byte/4*clk after the initial 16byte memory access [7]. To mitigate the delay from accessing other global SRAM cells NoC packets are prefetched before being processed. In this work we assume that all input operators are quantized to 8bits. In the final SpiNNaker2 chip the accelerator will support 16bit and 8bit input resolution and 32bit, 16bit and 8bit output resolution both in signed and unsigned variations. Further optional ReLU activation can be toggled. In the current version all other operations of deep networks like ReLU or max pooling are computed on the ARM core. By including the MAC array into PE, which increases silicon area by around 7% in 22nm FDX technology, the SpiNNaker2 becomes a competing-with-state-of-the-art DNN inference chip with a compute capacity of 4.6 TOPS and estimated energy efficiency of up to 6.4 TOPS/W [4].

3 MAPPING DNNS ON SPINNAKER2

We optimize our mapping of DNNs for low latency processing of single input samples. Therefore, layers are processed one after another on the whole chip, weights are fetched from DRAM, and intermediate feature maps are kept on-chip whenever possible. To reduce unnecessary data movement, original operations like CONV, activation function, memory padding, pooling, and quantization are fused into operation blocks executed on the same PE.

Due to limited PE SRAM and the highly-distributed nature of the given hardware system, full utilization of memory and processing units requires the splitting of resource-intensive tasks into independent pieces. The splitting is done on CONV, MM and matrix addition (MA), and other operations follow their resulting partition scheme. CONV layers are split first along output channels, then along width and height of ouput maps and finally along input channels, whereas MM is split along input width and then output width. There is no split priority for MA.

Furthermore the SpiNNaker2 architecture greatly benefits from hardware-aware meta-compiling of the task to avoid communication congestion and allows a faster and power-economical execution through data reuse and data locality strategies (see figure 2). We divided the system hierarchically into QPE-blocks each consisting of 4 QPEs. Furthermore, QPE-blocks were grouped into double

*Both authors contributed equally to this research.

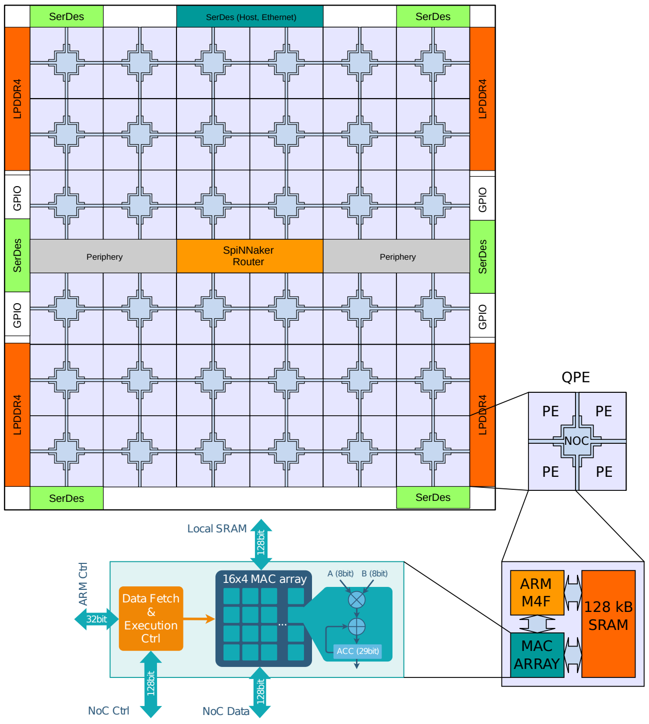

Figure 1: SpiNNaker2 chip overview

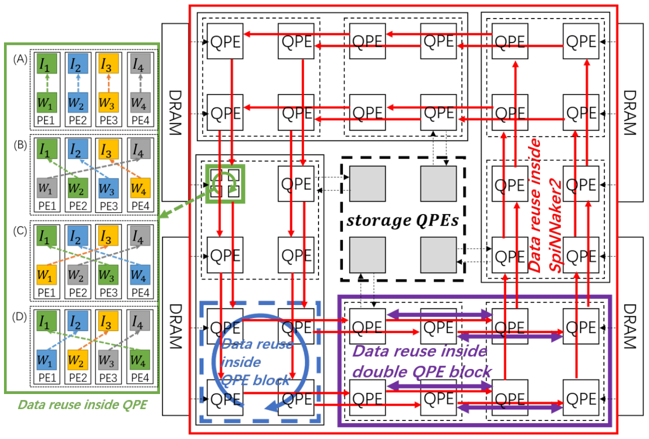
(A)
(B)
(C)
(D)
PE1 PE2 PE3 PE4
Data reuse inside QPE
DRAM
QPE
storage QPEs
Data reuse inside QPE block
Data reuse inside double QPE block
Data reuse inside SpiNNaker2

Figure 2: Data-reuse through NoC in SpiNNaker2

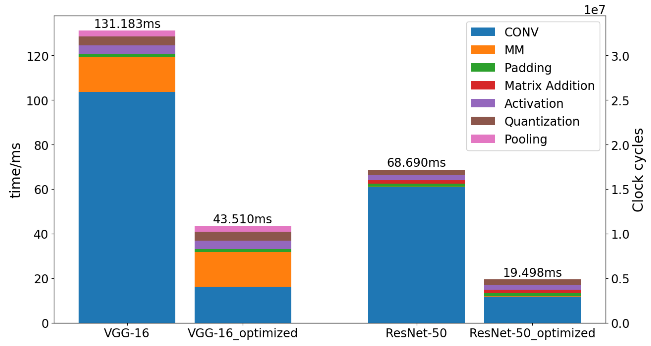

Figure 3: Overall processing time

QPE-blocks. Through data-reuse inside a QPE unit, one PE can fetch data from other PE SRAM cells. This can complete another 12 input-weights combinations in each QPE without having to fetch data from outside the QPE. This relaxes transmissions between QPE units and provides a significant increase of data bandwidth during computation. For data-reuse inside QPE-blocks, double QPE-blocks and inside the whole array, QPE units need to exchange data with each other and write into their local SRAM before computation. Via mapping into the hierarchical structure, data fetches are kept as local as possible, reducing transmission load on the NoC. Furthermore, the introduction of storage QPEs helps partly to overcome the bottleneck caused by the DRAM by keeping intermediate results on-chip. For CONV, after external fetch, only the weights flow via NoC inside SpiNNaker2 to ensure that each local part of the input feature map convolves with all the available on-chip weights. For the case of MM data reuse will decrease the overall workload of all MAC arrays and increase that of all ARM cores. Therefore only the storage QPE strategy is used for MM. The same applies for MA.

4 RESULTS

As the SpiNNaker2 chip is not yet available, we developed a Python simulator that replicates the timings of computation steps (ARM and MAC array) and data transfer (SRAM, NoC, DRAM). As benchmarks we simulated VGG-16 [10] and ResNet-50 model [3] with two different mapping strategies: The baseline method splits each layer into small pieces that can be computed in a single PE and reads the inputs and weights from DRAM for each piece. The optimized strategy employs the data reuse strategies shown in the previous section to reduce DRAM access. The inference time is shown in figure 3: For both models the optimized approach is more than 3 times faster than baseline and performs classification of single images in 43.5 ms resp. 19.5 ms. Note that the time used for fetching input/weights and writing results are included in CONV and MM. The simulation results were obtained with 500 MHz NoC frequency, 250 MHz PE clock and 2 GByte/s DRAM bandwidth.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 720270 and 785907 (Human Brain Project, HBP).

REFERENCES

[1] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.
[2] Steve Furber. 2016. Large-scale neuromorphic computing systems. *Journal of neural engineering* 13, 5 (2016), 051001.
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[4] Sebastian Höppner and Christian Mayr. 2018. SpiNNaker2-Towards Extremely Efficient Digital Neuromorphics and Multi-scale Brain Emulation. In *Proc. Neuro*

Inspired Comput. Elements Workshop. 1–21.

[5] Chen Liu, Guillaume Bellec, Bernhard Vogginger, David Kappel, Johannes Partzsch, Felix Neumärker, Sebastian Höppner, Wolfgang Maass, Steve B. Furber, Robert Legenstein, and Christian G. Mayr. 2018. Memory-Efficient Deep Learning on a SpiNNaker 2 Prototype. *Frontiers in Neuroscience* 12 (2018), 840. https://doi.org/10.3389/fnins.2018.00840

[6] Christian Mayr, Sebastian Hoeppner, and Steve Furber. 2019. SpiNNaker 2: A 10 Million Core Processor System for Brain Simulation and Machine Learning. *arXiv preprint arXiv:1911.02385* (2019).

[7] Bert Moons, Roel Uytterhoeven, Wim Dehaene, and Marian Verhelst. 2017. 14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 246–247.

[8] Michael Pfeiffer and Thomas Pfeil. 2018. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in neuroscience* 12 (2018).

[9] Teresa Serrano-Gotarredona, Bernabé Linares-Barranco, Francesco Galluppi, L Plana, and Steve Furber. 2015. ConvNets experiments on SpiNNaker. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2405–2408.

[10] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[11] Chetan Singh Thakur Thakur, Jamal Molin, Gert Cauwenberghs, Giacomo Indiveri, Kundan Kumar, Ning Qiao, Johannes Schemmel, Runchun Mark Wang, Elisabetta Chicca, Jennifer Olson Hasler, et al. 2018. Large-scale neuromorphic spiking array processors: A quest to mimic the brain. *Frontiers in neuroscience* 12 (2018), 891.

[12] Craig M Vineyard, Ryan Dellana, James B Aimone, Fredrick Rothganger, and William M Severa. 2019. Low-Power Deep Learning Inference using the SpiNNaker Neuromorphic Platform. In *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*. ACM, 13.