EasyChair Preprint

№ 9651

Performance Models and Energy-Optimal Scheduling of DNNs on Many-Core Hardware with Dynamic Power Management

Bernhard Vogginger, Florian Kelber,
Shambhavi Balamuthu Sampath, Johannes Partzsch and
Christian Mayr

February 2, 2023

Performance models and energy-optimal scheduling of DNNs on many-core hardware with dynamic power management

Bernhard Vogginger*, Florian Kelber*, Shambhavi Balamuthu Sampath*, Johannes Partzsch*†, Christian Mayr*†
*Chair of Highly-Parallel VLSI-Systems and Neuro-Microelectronics
Technische Universität Dresden
Dresden, Germany
Email: bernhard.vogginger@tu-dresden.de
†Centre for Tactile Internet with Human-in-the-loop (CeTI)
Technische Universität Dresden
Dresden, Germany

***Abstract*—Processing of deep neural networks (DNNs) at the edge may be limited by power or energy constraints of the used embedded hardware system. It is therefore desirable for the compiler to create efficient executables for given DNN models meeting the specific constraints. Here, we consider a low-power many-core hardware with 152 processing elements (PE), each containing an ARM M4F processor, 128 KB SRAM and a custom accelerator for DNN inference. Dynamic power management allows each core to switch between a high-speed and a low-power mode within tens of nanoseconds. For an energy-optimal parallelization of DNNs on the hardware, we first develop analytical performance models to predict the time and energy for executing a DNN layer with the custom accelerator. The models are fitted and validated using measurements on a prototype chip. In a second step we develop concepts for the energy-optimal parallelization of DNNs under latency constraints and evaluate them deploying the performance models: By dynamically switching between the operating modes more than $10\,\%$ of energy can be saved compared to the case running at high-speed mode only. The presented methodology and concepts are easily transferable to other many-core edge processors.**

***Index Terms*—deep neural networks, edge computing, many-core hardware, performance model, parallelization, power management**

I. Introduction

Dynamic Voltage and Frequency Scaling (DVFS) is an established technique for compute devices to control the speed and power of processing. By simultaneously decreasing the clock frequency and the supply voltage on a System-on-Chip (SoC), one can reduce the energy as the dynamic energy per operation scales with V^2 at cost of a lower speed. Very often, the DVFS settings are automatically adjusted by the operating system and not transparent to the user. In recent years, DVFS has been applied to DNN processing on ASICs [1], GPUs [2], and FPGAs [3]. While commonly DFVS is applied globally on the device level, one exception is the simulation study in [4] which performs fine-grained DVFS per processing element (PE) exploiting dynamic and static sparsity of DNNs for improved energy-efficiency.

Here, we consider the SpiNNaker2 many-core hardware offering fast PE-level power management with DVFS [5]. While originally conceived for the efficient simulation of biological neural networks, DNN accelerators have been integrated into the PEs making the system a powerful and energy-efficient device for DNN inference [6]. In this work we study the use of dynamic power management for an energy-optimal processing of DNNs on SpiNNaker2. This is crucial especially for edge applications where limited battery capacity and constraints on maximum power for the device are confronted with low-latency requirements for the real-time processing of sensor data.

The paper is structured as follows: Section II introduces the SpiNNaker2 many-core hardware. In Section III we develop energy and performance models of the DNN accelerator and validate them against hardware measurements. These models are deployed in Section IV where we suggest and evaluate power management concepts for optimal processing of CNNs on the hardware for different scenarios. We summarize the results in Section V.

II. Many-core hardware

The SpiNNaker2 hardware architecture is shown in Fig. 1). It consists of 36 Quad Processing Elements (QPEs), each in turn containing 4 PEs. The QPEs are interconnected on a 2D grid though a high-throughput Network-on-Chip (NoC), routing over 4 directions to each neighbor. This 2D mesh provides a faster means to share data space between PEs. Each PE contains an ARM M4F processor, 128 KB SRAM, and multiple accelerators focused on speeding up spiking and non-spiking neural network models. The machine learning accelerator (MLA) provides a 16x4 MAC array to speed up matrix multiplication and 2D convolution for 8bit or 16bit

The research leading to these results has received funding from the European Union's ECSEL Joint Undertaking under grant agreement n° 826655 - project TEMPO. This work has received funding from the EU Horizon 2020 framework under grant agreement 720270 (HBP SGA1), 785907 (HBP SGA2) and 945539 (HBP SGA3).

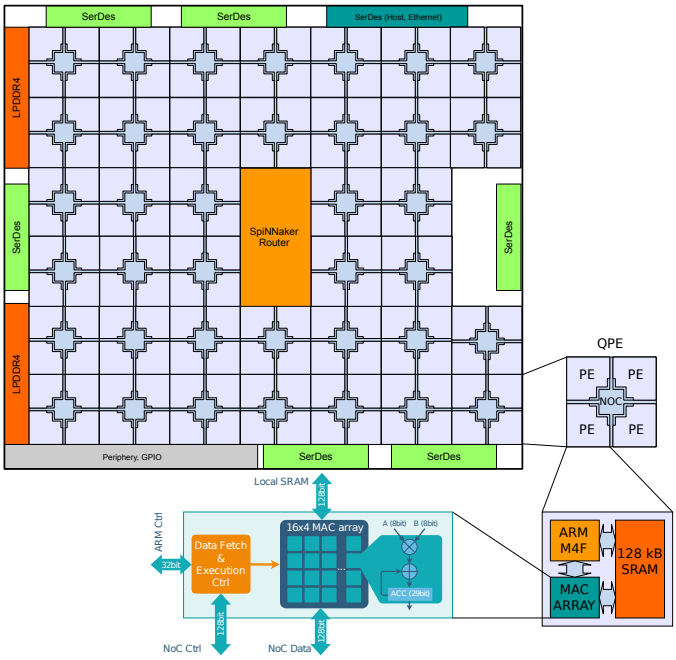

Fig. 1. SpiNNaker2 chip overview.

integers. The accelerator fetches data from the local SRAM and global NoC and writes the results back to the local SRAM. All 8bit unsigned MAC cells are reused by optional two's complement converter at the input and adder stage at the output to provide signed operation and a 16bit MAC. At the end the output is fed to optional post-processing modules which execute quantization and ReLU activation. An explanation of a more detailed data flow of an older version is provided in [7].

In the outermost NoC ring the chip provides a variety of periphery and additional interfaces, and access to a 2 GB LPDDR4 DRAM with 6.4 GB/s total throughput. Six serial chip-to-chip links and the SpiNNaker router allow the packet-based communication to other chips (not used in this study).

The system is specialized on fine-grained dynamic power control provided by 2 power lanes each PE can access, integrated adaptive body-biasing in 22FDX technology and 4 frequency dividers per QPE. While the supply voltages of the 2 power lanes for the PE logic (ARM core, accelerators) are set externally and globally per chip (typically between 0.45 V and 0.6 V), the clock frequencies can be chosen per PE between 50 MHz and 400 MHz. Each PE then can change to different performance levels (power lane and frequency combinations) during execution independent from other PEs or other NoC modules. This enables the system to adapt supply voltage and clock frequency to application requirements within tens of nanoseconds [5]. The SRAM is supplied with 0.8 V (fixed) and is accessed with the PE clock frequency. The NoC is typically operated at 300 MHz and 0.6 V (not changed during operation). Further details on the system architecture are available in [8].

III. Performance Models

a) Power measurements: We performed detailed measurements on a prototype chip with 2 QPEs [6], [8], where the MLA operation is looped 100k times while the power consumption is recorded for 3 power lanes: VDD08 for the SRAM, VDD06 for the NoC and periphery, and VDD04 for the PEs including the MLA. Execution times were measured with built-in timers of the ARM cores. Frequencies and voltages of the PE supply (VDD04) were swept to determine the optimal operation point by maximizing TOPS/W (see Fig. 2 for the convolution case). When increasing the frequency while keeping the voltage constant, the efficiency increases as the static power is drawn for a shorter time. The lower the supply voltage, the better is the efficiency at the same frequency due a smaller dynamic energy per operation. The higher the voltage, the higher is the maximum frequency where the MLA produces correct results.

From Fig. 2 we identify two specific operating points:

- **PL 1**: VDD04=0.5 V, f_{clk}=320 MHz for least energy
- **PL 2**: VDD04=0.6 V, f_{clk}=400 MHz for highest speed

further on called performance levels PL 1 and PL 2. PL 1 has the highest energy efficiency and thus represents the DVFS setting to accomplish tasks with least energy. Instead, PL 2 is the DVFS setting with the highest clock frequency with correct operation and thus provides the highest speed / lowest latency per task. No further PL is considered in this study as there are only two global power lanes for the voltage settings, and other frequencies at the selected voltages are less energy-efficient.

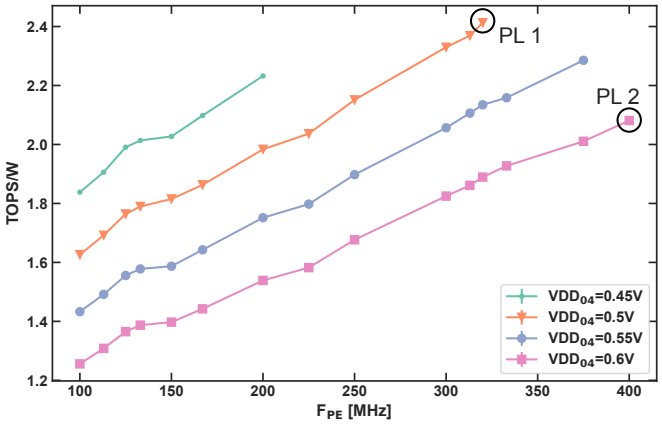

Fig. 2. Energy-efficiency for Conv2D operation on all 8 PEs of a SpiNNaker2 prototype chip for different voltage-frequency operation points.

b) Models: Based on the measurements, we have developed and fitted models to predict the energy consumption and time for the MLA. For sake of brevity, we focus on the performance models for conv2d mode of the accelerator. Similar models are available for matrix multiplication.

The energy is modelled as the sum of static and dynamic energy (1), where the static energy depends on the static power (leakage, clock enable etc.) over time (2). Instead, the dynamic

power is calculated by considering different atomic actions such as a memory accesses, cf. (3).

$$E_{\text{tot}} = E_{\text{dynamic}} + E_{\text{static}} \tag{1}$$

$$E_{\text{static}} = P_{\text{static}} \cdot t \tag{2}$$

$$E_{\text{dynamic}} = \sum_{actions} E_{\text{action}} \cdot N_{\text{action}} \tag{3}$$

Here, E_{action} denotes the energy per atomic action and N_{action} action count per application. The energy is modelled for the three different power supplies that are directly linked to different functional components: SRAM (VDD08), NoC (VDD06), and MLA (VDD04, this includes also the ARM core).

The *dynamic energy* for the different supplies is computed as follows:

$$E_{\text{dynamic,SRAM}} = E_{\text{read32}} \cdot N_{\text{read32}} + E_{\text{write32}} \cdot N_{\text{write32}} \tag{4}$$

$$E_{\text{dynamic,NoC}} = E_{\text{NoC-read}} \cdot N_{\text{NoC-read}} \tag{5}$$

$$E_{\text{dynamic,MLA}} = E_{\text{MLA-compute}} \cdot N_{\text{MLA-compute}} \tag{6}$$

The different actions are described in Table I.

The presented method for energy modeling is similar to Accelergy [9] by considering atomic actions with specific energies. However, we explicitly model static power whereas Accelergy uses idle operations. Our approach allows to predict the energy for different frequency settings without having to re-fit the parameters.

The time for running a convolutional layer with the machine learning accelerator depends on the parameter settings $\boldsymbol{x}_{\text{conv}}$, a set of fittable parameters $\boldsymbol{\theta}_{\text{conv}}$ and the clock frequency f_{clk}:

$$t_{\text{conv}} = t_{\text{conv}}(\boldsymbol{x}_{\text{conv}}, \boldsymbol{\theta}_{\text{conv}}, f_{\text{clk}}) \tag{7}$$

More concretely, we first model the number of clock cycles $\text{clks}_{\text{conv}}$:

$$\begin{aligned}\text{clks}_{\text{conv}} = \text{clks}_{\text{init}} + \left\lceil \frac{W_{\text{i}} - W_{\text{w}} + 1}{16} \right\rceil (H_{\text{i}} - H_{\text{w}} + 1) \\ \cdot (aW_{\text{w}}H_{\text{w}}C_{\text{i}} + \text{clks}_{\text{wb}}) \left\lceil \frac{C_{\text{o}}}{4} \right\rceil b \quad ,\end{aligned} \tag{8}$$

and then compute the total time:

$$t_{\text{conv}} = \frac{\text{clks}_{\text{conv}}}{f_{\text{clk}}} \quad . \tag{9}$$

Here, $\boldsymbol{x}_{\text{conv}}$ contains the width W_{i} and height H_{i} of the input feature maps, the number of input and output channels (C_{i} resp. C_{o}), and the width W_{w} and height H_{w} of the weight kernel, while $\boldsymbol{\theta}_{\text{conv}}$ contains four fittable parameters:

$$\boldsymbol{x}_{\text{conv}} = (W_{\text{i}}, H_{\text{i}}, C_{\text{i}}, C_{\text{o}}, W_{\text{w}}, H_{\text{w}}) \tag{10}$$

$$\boldsymbol{\theta}_{\text{conv}} = (\text{clks}_{\text{init}}, \text{clks}_{\text{wb}}, a, b) \tag{11}$$

The equation for the clock cycles in 8 was conceived based on the accelerator's operation workflow, see [7] for details. We remark that the execution time of the MLA for given parameter settings $\boldsymbol{x}_{\text{conv}}$ is not fixed, even not the number of clock cycles. As the weights are fetched using the NoC from another PE, both congestion in the NoC as well as concurrent SRAM accesses may slow down the operation. Hence, model fitting is needed for different scenarios and clock frequencies.

Parameter	Description
E_{read32}	Energy per SRAM read (32bit)
E_{write32}	Energy per SRAM write (32bit)
$E_{\text{NoC-read}}$	Energy per 128-bit NoC read within QPE
$E_{\text{MLA-compute}}$	Energy per MLA compute cycle (64 MAC ops)

TABLE I
PARAMETER DESCRIPTION

c) Parameter fitting: The time and energy models have been fit to the measurements on the SpiNNaker2 prototype chip. First, the static power for the various supply domains was extracted from measurements such as those in Fig. 2. Then, the time model parameters were obtained for 4 different convolutional layers with diverse settings. Finally, the energies per atomic operations (cf. Table I) were determined from the power measurements. For this, the action counts per execution of the convolutional layer were calculated based on the parameter setting $\boldsymbol{x}_{\text{conv}}$ according to the operating principle of the accelerator (formulas not shown). The energy and time models were determined for the two performance models PL 1 and PL 2. Note that for the NoC and the SRAM the static power and energies per action are the same for both performance levels.

Figure 3 shows the predicted energy for running the selected convolutional layers 100 000 times in the loop on 8 PEs in comparison to the measured energy at PL 2. One can see a very good match between the predicted and measured energy (relative error less than 5%), not only for the total energy but also for the energy per component. The error of the time models is less than 9% (not shown).

A dedicated Python package was developed for fitting the performance models and for their deployment in the next section. To obtain performance models at other operating points, the timing models have to be refitted. Instead, the energy models are specific to the underlying supply voltage and can be reused for all frequency settings.

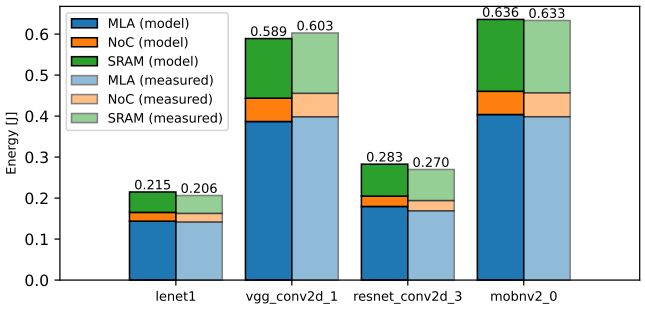

Fig. 3. Comparison of energy model and measurement for various convolution layers (100k loops on 8 PEs at PL 2).

IV. Power management concepts

In the following, we explore the power management capabilities of SpiNNaker2 for efficient DNN processing, choosing the performance level of the PE dynamically depending on the use case. Here we focus on processing convolutional layers, but we emphasize that the developed approaches equally apply to matrix multiplication on the MLA or software-based processing on the ARM cores. Note that we only present predictions for the SpiNNaker2 architecture. A hardware validation of the concepts and models is planned in the future using the final SpiNNaker2 chip.

a) Speed vs. energy for single convolutional layer: To run large convolutional layers on SpiNNaker2, they have to be split into smaller parts that fit into single processing elements. The main limitation is the available data memory per PE of 96KB, which is needed for both input feature maps and output feature maps. For example, the largest convolutional layer of the VGG-16 network for image classification [10] requires 3 MB each for input and output feature maps and 35 KB for the weights. The layer has to be split into many parts that even exceeds the number of PEs on the chip [11].

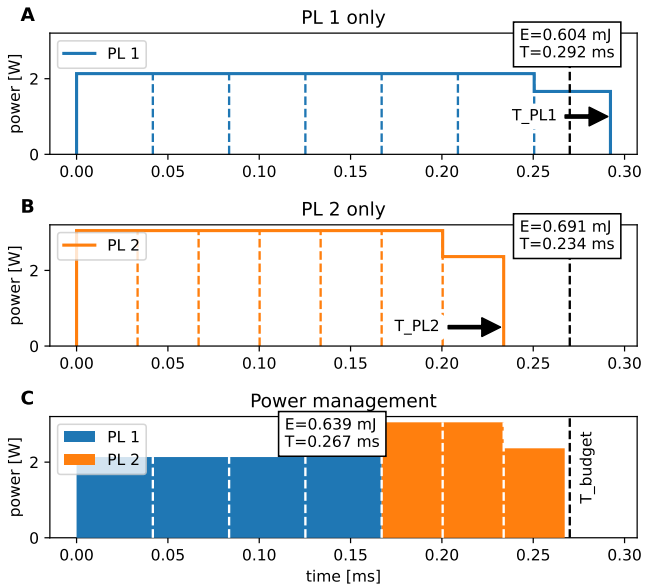

Fig. 4. Power consumption over time for processing VGG layer Conv-1-2 for different power management scenarios.

Figure 4 shows the estimated power over time to process the VGG layer Conv-1-2 on one SpiNNaker2 chip. The layer is split into 1024 parts so that in total 7 loops of processing are required. In the first 6 loops, all 152 PEs process one chunk of the layer in parallel, while in the last loop only 112 PEs are needed and the remaining cores go into sleep mode and only consume static power. When operated at performance level PL 1 (subfigure A), processing of the entire layer takes longer than at PL 2 (subfigure B). However, both the total energy and the maximum consumed power are less for PL 1. This shows the general flexibility to choose between speed and energy efficiency for processing DNNs on SpiNNaker2.

b) Power management scenarios: There are different application scenarios where power management, i.e., switching between performance levels, can be applied. For all scenarios we assume that there are exactly two performance levels. All PEs are enabled all the time, but can go into sleep mode to consume only static power. We have identified the following scenarios:

1) **Speed**: DNN should be processed as fast as possible.
2) **Energy**: DNN should be processed with least energy.
3) **Limited time**: Time for processing the DNN is limited, e.g., when images arrive at a certain frame rate.
4) **Limited power**: The power the chip can draw is limited, e.g., by the power supply unit.

The first two scenarios don't require any specific power management, either the higher or lower performance level is chosen. Instead, for the other scenarios an active power management strategy is needed to operate the chip at optimal energy efficiency under time or power constraints. In the following, for sake of brevity, we focus on approaches for limited time.

c) Limited time budget: single layer: We assume that the time to process a single convolutional layer is limited and that the available time budget T_{budget} is between the time needed using PL 1 or PL 2 only. Of course, one can use PL 2 only to meet the time constraint. However, by processing parts of the layer at the lower performance level PL 1, one can save energy while still not exceeding the time limit. Figure 4C shows an example applying such active power management for the same VGG layer as before: The first 4 loops are processed at PL 1 and the last 3 loops at PL 2. The computation of the layer stops shortly before the assumed time budget of $0.27\,\mathrm{ms}$. $7.4\,\%$ less energy is needed compared to using PL 2 only. The number of loops to process at PL 1 was determined automatically. With the suggested approach and for the considered layer, one can reduce the energy by up to $11\,\%$ depending on the given time budget.

d) Limited time budget: network: Similarly, the power management can be applied on the network level. Let's assume that all 13 convolutional layers of VGG-16 need to be processed within $3.3\,\mathrm{ms}$ as shown in Fig. 5: When processed at PL 1, time to finish is $3.6\,\mathrm{ms}$ and hence too long. Note that the time and average power per layer varies depending on the parameters of the layer. At PL 2, the processing finishes more than $0.3\,\mathrm{ms}$ before the end of the time budget. This remaining time can be used to process some layers at the lower performance level to minimize the total energy. The bottom plot in Fig. 5 shows the optimal solution that finishes just before the time limit and requires the least possible total energy, which saves $6.6\,\%$. Depending on the given time budget, up to $12\,\%$ of energy can be saved in comparison to using PL 2 only.

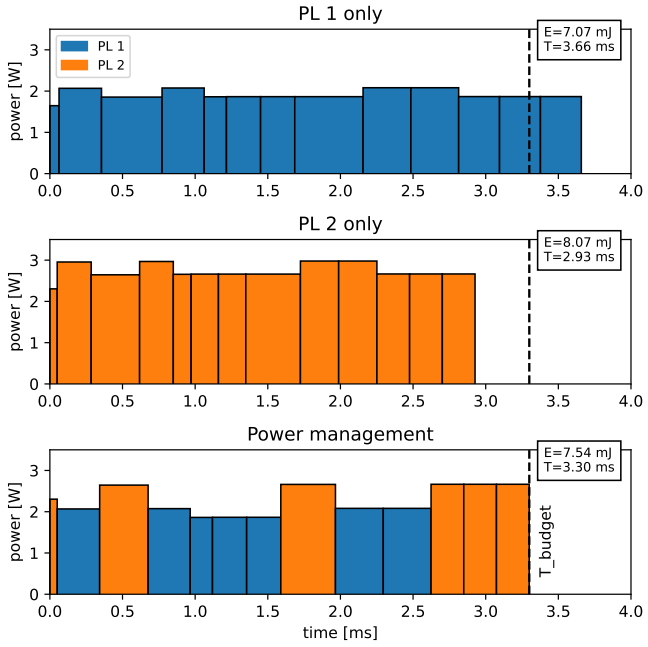

Fig. 5. Energy-optimized operation of all 13 convolutional layers of VGG-16. The width and height of each vertical bar denote the time and average power per layer.

V. Conclusion

We have developed grey box time and energy models for the machine learning accelerator of the SpiNNaker2 many-core hardware. Compared to others, our methodology considers also static power and inherently supports changing the clock frequency. In a second step, power management concepts are proposed that make use of the fine-grained DVFS available on the many-core hardware. It was shown for scenarios with limited processing time that, by switching the performance level for individual layers or parts of layers, the total energy can be reduced by up to 12% compared to using the high-speed performance level only. Further savings are expected for the final SpiNNaker2 chip due to a reduced static power. The concepts are easily transferable to other many-core architectures with fine-grained power management.

In the future, both the performance models and the power management strategies shall be integrated into the DNN compiler for SpiNNaker2. This will allow to find an energy-optimal mapping and scheduling for entire networks.

References

[1] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "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)*, 2017, pp. 246–247.

[2] Z. Tang, Y. Wang, Q. Wang, and X. Chu, "The impact of gpu dvfs on the energy and performance of deep learning: An empirical study," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, 2019, pp. 315–325.

[3] W. Jiang, H. Yu, J. Zhang, J. Wu, S. Luo, and Y. Ha, "Optimizing energy efficiency of cnn-based object detection with dynamic voltage and frequency scaling," *Journal of Semiconductors*, vol. 41, no. 2, p. 022406, 2020.

[4] S. Liu and A. Karanth, "Dynamic voltage and frequency scaling to improve energy-efficiency of hardware accelerators," in *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 2021, pp. 232–241.

[5] S. Höppner, B. Vogginger, Y. Yan, A. Dixius, S. Scholze, J. Partzsch, F. Neumärker, S. Hartmann, S. Schiefer, G. Ellguth, L. Cederstroem, L. A. Plana, J. Garside, S. Furber, and C. Mayr, "Dynamic power management for neuromorphic many-core systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 8, pp. 2973–2986, 2019.

[6] Y. Yan, T. C. Stewart, X. Choo, B. Vogginger, J. Partzsch, S. Höppner, F. Kelber, C. Eliasmith, S. Furber, and C. Mayr, "Comparing loihi with a SpiNNaker 2 prototype on low-latency keyword spotting and adaptive robotic control," *Neuromorphic Computing and Engineering*, vol. 1, no. 1, p. 014002, jul 2021. [Online]. Available: https://doi.org/10.1088/2634-4386/abf150

[7] S. M. A. Zeinolabedin, F. M. Schüffny, R. George, F. Kelber, H. Bauer, S. Scholze, S. Hänzsche, M. Stolba, A. Dixius, G. Ellguth, D. Walter, S. Höppner, and C. Mayr, "A 16-channel fully configurable neural soc with 1.52 μW/Ch signal acquisition, 2.79 μW/Ch real-time spike classifier, and 1.79 TOPS/W deep neural network accelerator in 22 nm fdsoi," *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–1, 2022.

[8] S. Höppner, Y. Yan, A. Dixius, S. Scholze, J. Partzsch, M. Stolba, F. Kelber, B. Vogginger, F. Neumärker, G. Ellguth, S. Hartmann, S. Schiefer, T. Hocker, D. Walter, G. Liu, J. Garside, S. Furber, and C. Mayr, "The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing," 2021. [Online]. Available: https://arxiv.org/abs/2103.08392

[9] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] F. Kelber, B. Wu, B. Vogginger, J. Partzsch, C. Liu, M. Stolba, and C. Mayr, "Mapping deep neural networks on spinnaker2," in *Proceedings of the Neuro-inspired Computational Elements Workshop*, 2020, pp. 1–3. [Online]. Available: https://easychair.org/publications/preprint/JPPG