A Heterogeneous Multi-Core System-on-Chip for Energy Efficient Brain Inspired Computing

A Heterogeneous Multi-Core System-on-Chip for Energy Efficient Brain Inspired Computing

Antonio Pullini*, Francesco Conti*†, Davide Rossi†, Igor Loi†, Michael Gautschi*, Luca Benini*†
*Integrated Systems Laboratory, ETH Zürich, Zurich, Switzerland †DEI, University of Bologna

***Abstract*— Convolutional Neural Networks (CNNs) have revolutionized computer vision, speech recognition and other fields requiring strong classification capabilities. These strenghts make CNNs appealing in edge node internet-of-things (IoT) applications requiring near-sensors processing. Specialized CNN accelerators deliver significant performance per watt and satisfy the tight constraints of deeply embedded devices, but they cannot be used to implement arbitrary CNN topologies or non-conventional sensory algorithms where CNNs are only a part of the processing stack. A higher level of flexibility is desirable for next generation IoT nodes. Here we present *Mia Wallace*, a 65nm System-on-Chip integrating a near-threshold parallel processor cluster tightly coupled with a CNN accelerator: it achieves peak energy efficiency of 108 GMAC/s/W @ 0.72V and peak performance of 14 GMAC/s @ 1.2V, leaving 1.2 GMAC/s available for general-purpose parallel processing.**

***Keywords*—Convolutional Neural Networks, Multi-Processor System On a Chip, Near Threshold Computing, Heterogeneous Computing**

I. Introduction

Convolutional networks are becoming increasingly popular in computer vision thanks to their outstanding accuracy and generalization capability in object detection, scene parsing, and image segmentation tasks [1]. As CNNs are computationally expensive, they are typically deployed in high-performance, power-hungry servers "in the cloud" and not on embedded devices. However, the ability to compress low information density data into a highly informative compressed state (e.g. a classification tag) is attractive also for low-power embedded devices. For example, smart visual sensor nodes could exploit it to minimize the amount of energy spent in data transmission, by sending to the cloud only classification tags or pre-classified data. ASIC accelerators are the standard way to cope with significant workloads in low-power embedded devices, but for the specific task of CNNs they lack the flexibility to adapt to the great variety of different topologies. Moreover, they are limited in scope to a scenario where CNNs are the sole application run on the node.

This work extends the previous abstract by Pullini et al. [2]; we propose a 65nm energy-efficient system-on-chip based on a hybrid HW/SW approach to CNN acceleration. We rely on a near-threshold parallel platform featuring four single-issue OpenRISC cores enhanced for efficient fixed point computations and a hardware accelerator for convolution-accumulation operations, which constitute the bulk of the computational load of CNNs. The proposed approach joins the flexibility of software-programmable processors with the performance and energy efficiency boost of specialized hardware, suitable for a new generation of IoT applications based on brain inspired computing.

A common way to accelerate CNNs on programmable hardware relies on GP-GPUs, which are able to reach extremely high throughput (up to 6 Top/s), but consume tens or hundreds of Watts [3][4]. Embedded CNN implementations on platforms such as ODROID-XU [5] or CEVA [6] provide tens of Gop/s within a power budget of a few watts. Movidius Myriad 2 has 12 8-way VLIW SHAVE processors, claimed to be working at 600MHz within a power envelope of $0.5\,\mathrm{mW}$ [7], for a total of up to 120 Gop/s/W. Despite its very high claimed peak efficiency, it is not a direct point of comparison for our work as it targets more powerful embedded systems such as smartphones, UAVs with a power envelope more than $10\times$ higher than that of Mia Wallace.

Low-power application specific CNN accelerators often focus on convolutional layers as they dominate CNNs. Origami is a convolutional accelerator providing a peak energy efficiency of 803 GOPS/W in 65nm technology [8]. Although this solution is efficient, it requires additional components at system level to implement full CNNs. Other hardware solutions implement a whole CNN including pooling, activation layers and fully-connected layers exploiting different computational models and architectures. A reconfigurable dataflow architecture Neuflow was presented in IBM 45nm SOI technology with a throughput up to 1280 Gop/s and an core energy efficiency of 490 Gop/s/W [9]. ShiDianNao presented an architecture exploiting the 2D structure of the CNN reaching 128 Gop/s and 400 Gop/s/W energy efficiency [10]. Eyeriss includes an array of 14x12 reconfigurable processing elements connected through a network-on-chip. It reduces data movements and exploit data re-use compression to reduce I/O bandwidth [11]. Jaehyeong et al. present a DNN processor that uses an image tiling scheme for reducing off-chip memory access and an algorithmic approach (Principal Component Analysis) to reduce the dimension of the kernels [12]. Different approaches to low-power sensor data analytics have also been recently explored with promising results, using e.g. convolutional deep

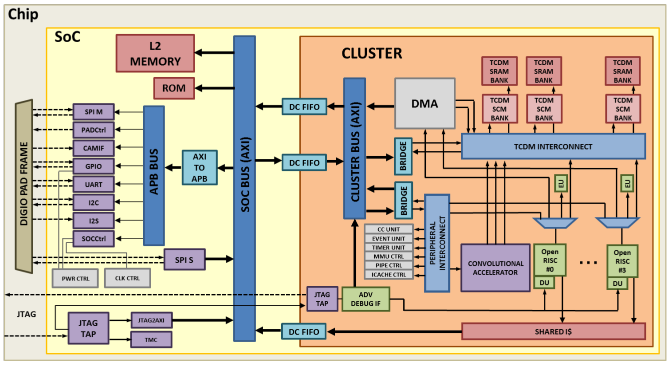


Fig. 1: *Mia Wallace* SoC architecture

belief networks (DBN) instead of CNNs. Park et al. [13] report up to 1939 Gop/s/W when combining DBN learning and inference.

Contrasted to most of the presented CNN accelerators, *i*) our proposal is a full multicore heterogeneous SoC, working in a SW-driven fashion (in fact, the CNN accelerator accounts only for 15% of the full cluster area); *ii*) we focus on a technique for sustainable usage of the available memory bandwidth, which is often the scarcest resource in CNN computations; *iii*) the flexible architecture we propose can combine CNNs with many other sensor data analysis techniques, using SW cores and the CNN accelerator concurrently.

II. SoC Architecture

The proposed SoC implements the third generation PULP (Parallel Ultra-Low-Power) platform[1] extended with a dedicated accelerator for convolution intensive processing [16]. The programmable computing engine of the SoC is based on a tightly coupled cluster of 4 OpenRISC ISA cores called OR10N enhanced for energy efficient digital signal processing [17]. The cluster features a shared 4kB latch-based Standard Cell Memory (SCM) [18] instruction cache that, coupled with a private per-core L0 buffer, increases energy efficiency by 30% with respect to an SRAM-based private cache architecture [15]. The cores share an explicitly managed Tightly Coupled Data Memory (TCDM). The TCDM features 8 word-level interleaved banks connecting the processors through a non-blocking interconnect to minimize banking conflict probability. Each logical bank is implemented as a heterogeneous memory, composed of 64kB of SRAM banks and 8kB of SCM banks; by disabling SRAMs in the cluster entirely, it is possible to extend the operating range well below the limits imposed by SRAM scaling (down to 0.62V in the case of *Mia Wallace*).

A multi physical-channel DMA enables fast and flexible communication with 256kB of L2 memory. The set of peripherals available on the SoC include: 200 Mbit/s SPI interfaces (master/slave and single/quad mode), I2C, 50 Mbit/s I2S, GPIOs, bootup ROM and JTAG interface for debug and test purposes. To provide high energy efficiency across a wide range of workloads, the cluster and the rest of the SoC are in different clock and voltage domains, isolated by dual-clock FIFOs and level shifters. Fine-grained tuning of the SoC and cluster frequencies is performed by two Frequency-Locked Loops [19].

A dedicated Hardware Convolution Engine (HWCE) extends the cluster to efficiently implement convolve-accumulate operations. The "deep core" of the HWCE are two sum-of-products (SoP) units, providing a peak throughput of 2 5×5 convolutions per cycle on 16-bit inputs. The 5×5 SoP enables native computation of the great majority of CNNs, which use 5×5 and 3×3 filters [20][21]. The accelerator is integrated in the cluster as a triple of masters of the TCDM interconnect, using the same interconnect as the processor cores. Offload of HW accelerated tasks is performed through a dedicated slave port mapped on a peripheral interconnect, where a shadow configuration register enables asynchronous control of the HWCE without stopping its execution.

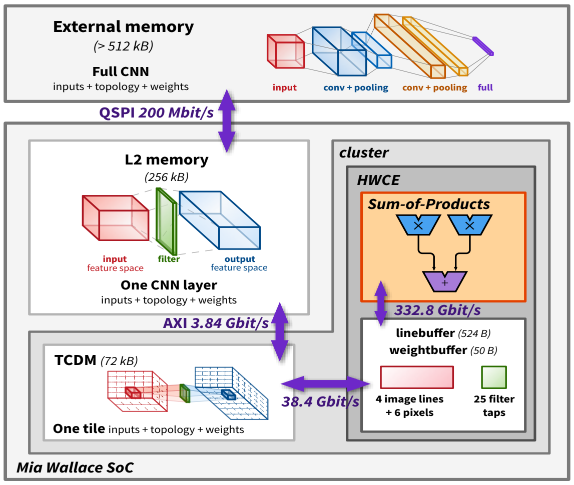


Fig. 2: Memory hierarchy and CNN application hierarchy.

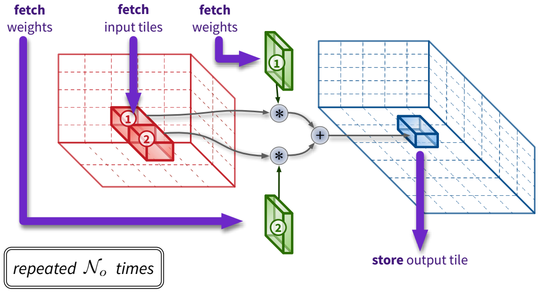


Fig. 3: Schematization of a single iteration of a tiled convolutional layer with the strategies proposed in Section III.

III. CNNs on Mia Wallace

A first challenge in implementing CNNs in a small embedded device such as Mia Wallace is the CNN working set size, forcing to maximize usage of local memory (the TCDM) and minimize data exchange with other levels of the memory hierarchy. Deep CNNs are naturally divided in *layers* (e.g. convolutional, pooling and fully-connected). Each convolutional layer builds a 3D space of output feature maps from a 3D space of input feature maps, using a matrix of convolutional filters (see e.g. [21]); typically, even a single layer is too big to be stored entirely in the 72 kB TCDM.

This challenge can be addressed noting that CNNs have a hierarchical structure that maps nicely to the explicitly managed memory hierarchy of *Mia Wallace*. We assume that the CNN topology (inputs and weights for the full network) resides on an external memory accessible via QSPI (see Figure 2). The application brings one layer at a time to the L2 memory via QSPI. Since the 72 kB TCDM cannot typically host inputs, weights and outputs for a whole layer, it is necessary to further split the workload and the work set in independent chunks that are executed one after the other from TCDM. This can be achieved by *tiling*, i.e slicing the input space in a $\mathcal{N}_i \times \mathcal{H} \times \mathcal{W}$ grid (in the feature, height and width dimensions); the output space in a $\mathcal{N}_o \times \mathcal{H} \times \mathcal{W}$ grid; and the filter space in a $\mathcal{N}_i \times \mathcal{N}_o$

[1] The first generation PULP architecture is presented in [14], while the second generation is presented in [15]. Further information regarding the PULP platform can be found in the project web page http://www.pulp-platform.org.

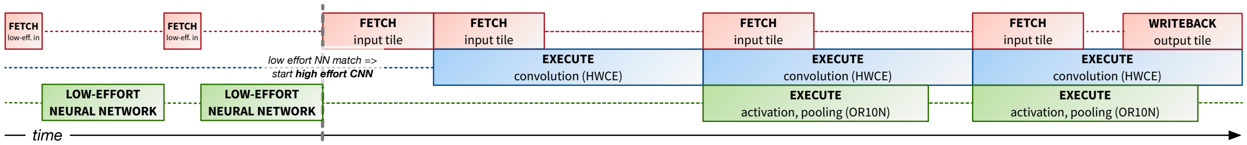


Fig. 4: SW pipeline with low-effort NN run in SW that triggers a high-effort CNN run by SW, HWCE and DMA cooperation.

grid. The computation is performed on one tile (kept in L1) at a time.

Each output tile is computed as a sum of contributions from a set of $\mathcal{N}_i$ input tiles, but from the tiling perspective there is significant freedom on how to organize this accumulation. Using the tiling strategy shown in Figure 3, for each output tile the contribution of all $\mathcal{N}_i$ related input tiles is computed sequentially; then the next output tile is computed. This requires to reload each input tile up to $\mathcal{N}_o$ times. The transfer overhead can be reduced by minimizing $\mathcal{N}_i$; in the limit where there is a single input tile ($\mathcal{N}_i = 1$), it is no longer necessary to reload tiles multiple times. The flexible architecture of *Mia Wallace* enables this strategy, without being limited to it.

The presence of the HWCE relieves the OR10N cores of the heavy task of computing convolutions in CNN, substituting it with the much lighter task of DMA and HWCE control. In both the DMA and HWCE it is possible to enqueue a set of jobs (up to 8 for the DMA, 2 for the HWCE), after which the cores are free to perform other useful work. For example, it is possible to establish a software pipeline where the cores execute activation and pooling of the previous layer while the HWCE works on the "bulk" convolution of the current layer. It is also possible to introduce double buffering to hide data transfer overheads.

An additional advantage of the availability of both SW cores and a HW accelerator is that of establishing a trigger for a high-effort CNN execution with a low-effort, low-power first stage such as a shallow simplified non-convolutional neural network executed in SW [22], which *Mia Wallace* can execute in pure SW, as shown in Figure 4. Thanks to its reduced memory requirements, the LE-NN can also be run using only the SCM portion of the TCDM, making it possible to reach the minimum supply voltage of 0.62V and minimize the overall power envelope. Section IV evaluates the maximum workload of such a kind that can be supported in *Mia Wallace*, and the relative energy efficiency.

IV. Results

In this Section, we evaluate performance and efficiency of our platform on the manufactured *Mia Wallace* prototype chips[2]. Figure 5 shows a microphotograph of one of the chips, along with its main features and parameters such as working operating frequecies and power range. The total size of *Mia Wallace* is $3.95\,\mathrm{mm} \times 1.88\,\mathrm{mm}$.

We measured the baseline peak efficiency that can be expected from the platform on CNNs by letting the HWCE run while the rest of the cluster is silent and there is no data transfer. The average throughput in this case is 36.5 MAC/cycle - in other words, 1.46 5×5 convolutions per cycle. To put this value into perspective, a high-accuracy CNN architecture such as GoogLeNet [20] requires a total of 2.45×10^9 MAC operations when applied on a 320×240 input image (a realistic image size for low-power camera sensors); at peak, our platform could sustain a similar computational workload in real time. At $0.72\,\mathrm{V}$, pure HWCE execution reaches 3.53 GMAC/s of throughput within $15\,\mathrm{mW}$, for an overall 236 GMAC/s/W of overall efficiency.

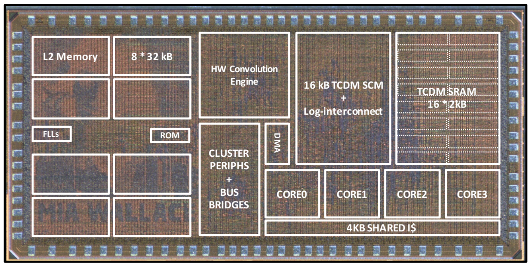
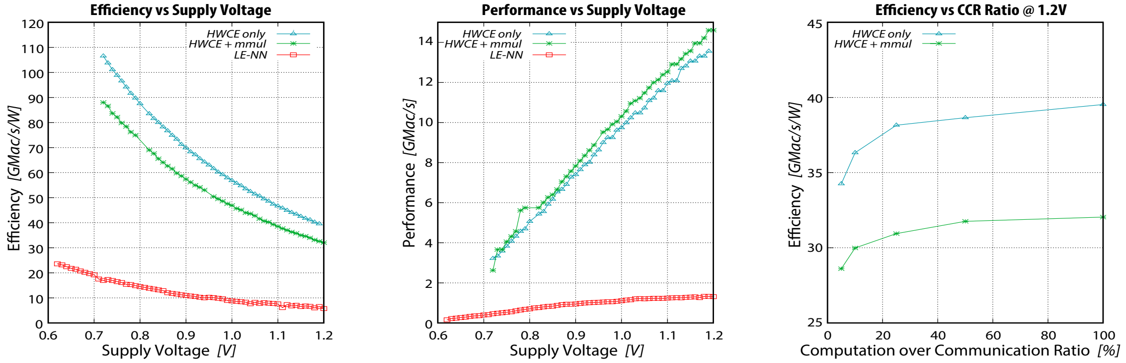


Fig. 5: *Mia Wallace* chip micrograph and main features.

Sharing memory between software cores and the HWCE introduces the opportunity to apply also activation and pooling to the output set of the accelerator directly in-place in the cluster, by software. These operations are typically simple, and they show a high degree of variability between different CNN topologies (e.g., max- and avg-pooling on different sizes, different types of nonlinear activations); hence, they are not good candidates for hardware acceleration. The shared-memory acceleration technique allows to implement these kernels without any additional performance/energy overhead to move data from the accelerator. For example, at $0.72\,\mathrm{V}$ 2×2 max-pooling has a cost of $\sim$3.8 cycles and $\sim$$640\,\mathrm{pJ}$ per input pixel for computation on 4 SW cores; if this data had to be copied from a private HWCE memory to L2 and then to the shared TCDM before being used, there would be a small time penalty ($\sim$ 0.5 cycles per input pixel), and a significant energy overhead ($250\,\mathrm{pJ}$ per pixel - a 39% increase). If the only non-linearity applied to output pixels is a simple ReLU, then the energy overhead of data movement from private accelerator memories becomes even more expensive - while data movement overheads are similar to the max-pooling case, pure computation costs only 0.7 cycles per pixel and $120\,\mathrm{pJ}$; data movement would cost more than $3\,\times$ computation on such a small kernel.

As mentioned in Section III, the most critical aspect of executing CNNs on an embedded platform such as PULP is data transfer to/from the local memory, which requires tiling. As a way to understand how well convolutional layer execution

[2] All tests are run using similar uniformly distributed random data sets for inputs and weights; as many CNNs work on sparse data in the latter stages and on dense data in the first stage, this provides a reasonable upper bound on power consumption.



(a) Energy efficiency of conv layer for *HWCE*, *HWCE+MMUL* and *LE-NN* while varying V_{DD}.

(b) Throughput of conv layer for *HWCE*, *HWCE+MMUL* and *LE-NN* while varying V_{DD}.

(c) Efficiency of conv layer for *HWCE* and *HWCE+MMUL* while varying computation over communication ratio.

Fig. 6: Energy efficiency and throughput of convolutional layer execution on *Mia Wallace* in the *HWCE*, *HWCE+MMUL* and *LE-NN* tests.

		Platform & Tech.	*Conv Perf.* [GMAC/s]	*Power*[a] [mW]	*Conv Efficiency* [GMAC/s/W]	*Conv Layer*	*Pool Layer*	*Dense Layer*	*Fully SW Programmable*	*I/O*
SW	NVIDIA Fermi [3]	GPU 22nm	3.2k	250k	12.7	✓	✓	✓	✓	PCIe
	NVIDIA Tegra [3]	GPU 28nm	84	11k	7.6	✓	✓	✓	✓	mobile[b]
	ODROID-XU [5]	CPU 28nm	1.21	2.8k	0.43	✓	✓	✓	✓	mobile[b]
	STM32 L476 [23]	uC	0.026	10	2.6	✓	✓	✓	✓	uC[c]
ASIC	NeuFlow [9]	45nm	147	600	245[d]	✓	✓	✗	✗	DDR[d]
	ShiDianNao [10]	65nm	64	320	200	✓	✓	✓	✗	-
	Origami [8]	65nm	27.5	93	296	✓	✗	✗	✗	parallel
	Jaehyeong et al. [12]	65nm	32	45	710[e]	✓	✓	✗	✗	-
	Eyeriss [11]	65nm	23	278	83	✓	✗	✗	✗	-
MPSoC	***Mia Wallace* SoC@0.72V**	**65nm**	**3.3**	**31**	**108**	✓**(HW)**	✓	✓	✓	**uC[c]**
	Mia Wallace HWCE@0.72V			11	300					
	***Mia Wallace* SoC@1.2V**	**65nm**	**14**	**359**	**39**	✓**(HW)**	✓	✓	✓	**uC[c]**
	Mia Wallace HWCE@1.2V			127	110					

[a] For ASICs and *Mia Wallace* power and efficiency numbers refer to core power, excluding I/Os.
[b] Includes I/O interfaces typically found in mobile devices, such as USB, Wi-Fi, and ones such as SPI, I2C that are usually present in mobile SoCs.
[c] Includes typical microcontroller I/O interfaces such as SPI, I2C, I2S.
[d] Data collected post-synthesis and thus not considering power overheads from backend (e.g. from the clock tree).
[e] Weights produced on-chip from a small set of PCA bases to save area/power. No evaluation on the general validity of this approach is presented in [12], beyond the reported MNIST use case.

TABLE I: Comparison between *Mia Wallace* and several platforms representative of the state-of-the-art in CNN inference.

can be superimposed to data transfer and other computations (e.g. subsampling in a pooling layer), we define *computation-to-communication ratio* (CCR) metric as the ratio between executed MAC operations and bytes transferred in/out of the cluster TCDM. We swept CCR from 5 (i.e. full overlap between busy time of DMA and HWCE) to 100 (i.e. the DMA is active only for a fraction of the computation time). We performed two kinds of CNN tests: in *HWCE* tests one OR10N core is used only for controlling the execution and the other three are idle, while in *HWCE+MMUL* tests the cores are used for a high effort computation (a matrix multiplication), a worst case where the cores are used in a software pipeline. Finally, a third test (*LE-NN*) represents the continuous running of a small fully-connected artificial NN such as that described in Section III.

Figures 6a and 6b report respectively the energy efficiency in GMAC/s/W and the throughput in GMAC/s when executing the tests; Figure 6c shows the efficiency variation at the 1.2 V operating point when sweeping the CCR. The results of the *LE-NN* tests indicate that at 0.62 V, power is below 6.3 mW at 38 MHz, and the platform can support a LE-NN of up to 150 MMAC/s, which is enough to run a very small shallow non-convolutional neural network. Conversely, a pure SW implementation of CNNs would be too slow to run a state-of-the-art CNN with more than 10^9 MACs on an embedded SoC. The HWCE solves part of this problem, delivering good results even when the cost of data transfers in energy and in increased memory contention is highest. For example, when the CCR is 5 the energy efficiency is still as high as 91 GMAC/s/W. In the more common case of a relatively high CCR, *Mia Wallace* reaches an even better overall efficiency of 108 GMAC/s/W at 0.72V, or 9.26 pJ per MAC.

When we consider the *HWCE+MMUL* tests, the cooperative execution on both the OR10N cores and the HWCE provides a 10% performance boost. Counting two operations per MAC, Figure 6 shows that a compound CNN workload of up to 30 Gop/s can be sustained by the *Mia Wallace* platform. Net HWCE throughput differs by less than 5% from the performance in the *HWCE* tests - superposition of HWCE work with a significant SW load does not hit CNN performance. Supporting additional workload benefits the overall CNN execution in several ways, not only by directly improving its throughput. For example, pooling layers are pure reductions: executing one in the software pipeline improves the CCR by performing more operations and even more by reducing the amount of data to be written back to L2, improving overall throughput in turn. This key consideration helps understanding the *HWCE+MMUL* results in Figure 6: while there is of course a small net power cost in having both the cores and the HWCE run at the same time, additional computation improves the CCR by both raising the number of ops and, in the case of pooling, by reducing the amount of data to be moved, making the effective efficiency loss almost negligible.

Table I provides a final summary of our contribution and its positioning with respect to the state-of-the-art in CNN inference executed both in SW and in HW accelerators. In terms of energy efficiency, the HWCE in *Mia Wallace* achieves results comparable to state-of-the-art dedicated ASICs [9][10][8][11], and the efficiency of the full *Mia Wallace* cluster, including cores, DMA and interconnects, is still in a similar range. Differently from most other platforms, *Mia Wallace* is able to execute full CNNs of arbitrary size using the methodology illustrated in Section III (whereas except for [10] ASICs are mostly limited to conv and pool layers). Moreover, as opposed to all ASIC architectures compared in the table, it can also execute arbitrary code using CNNs as part of more complex pipelines.

V. Conclusion

This work presented *Mia Wallace*, a 65nm system-on-chip targeting the emerging class of near-sensor IoT applications. We have shown how the flexible architecture of *Mia Wallace* can be exploited to efficiently run Convolutional Neural Networks, which provided state-of-the-art results in visual, audio and signal classification and would be highly desirable in a IoT scenario. On the one hand, this allows the execution of a low-performance, low-effort SW neural network within 6.3 mW of overall power budget. On the other hand, the execution of a CNN convolutional layer can be superimposed with additional workload, for a maximum compound average throughput of ∼30 GOP/s at 400 MHz.

Acknowledgment

This work was funded by the Swiss National Foundation under grant 162524 (MicroLearn: Micropower Deep Learning), armasuisse Science & Technology and the ERC MultiTherman project (ERC-AdG-291125).

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1106–1114.

[2] A. Pullini, F. Conti, D. Rossi, I. Loi, M. Gautschi, and L. Benini, "A heterogeneous multi-core system-on-chip for energy efficient brain inspired vision," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 2910–2910.

[3] L. Cavigelli, M. Magno, and L. Benini, "Accelerating real-time embedded scene labeling with convolutional networks," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 108:1–108:6.

[4] S. Chintala, "convnet-benchmarks," 2016. [Online]. Available: https://github.com/soumith/convnet-benchmarks

[5] F. Conti, A. Pullini, and L. Benini, "Brain-inspired Classroom Occupancy Monitoring on a Low-Power Mobile Platform," in *CVPR 2014 Workshops*, 2014.

[6] "CEVA-MM3101 - Programmable, Low Power Imaging and Vision Platform for Camera-Enabled Devices," http://www.ceva-dsp.com/CEVA-MM3101.

[7] Movidius, "Ins-03510-c1 datasheet," 2014, datasheet of Myriad 2 Vision Processor. [Online]. Available: http://uploads.movidius.com/1441734401-Myriad-2-product-brief.pdf

[8] L. Cavigelli and L. Benini, "Origami: A 803 GOp/s/W Convolutional Network Accelerator," *arXiv:1512.04295 [cs]*, Dec. 2015.

[9] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[10] Z. Du *et al.*, "Shidiannao: Shifting vision processing closer to the sensor," in *ISCA '15*, ser. ISCA '15. New York, NY, USA: ACM, 2015, pp. 92–104. [Online]. Available: http://doi.acm.org/10.1145/2749469.2750389

[11] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *ISSCC-16*, Jan 2016, pp. 262–263.

[12] J. Sim *et al.*, "A 1.42tops/w deep convolutional neural network recognition processor for intelligent ioe systems," in *ISSCC 2016*, Jan 2016, pp. 264–265.

[13] S. Park, K. Bong, D. Shin, J. Lee, S. Choi, and H.-J. Yoo, "A 1.93TOPS/W Scalable Deep Learning/Inference Processor with Tetra-Parallel MIMD Architecture for Big-Data Applications," in *Solid- State Circuits Conference - (ISSCC), 2015 IEEE International*, 2016.

[14] D. Rossi *et al.*, "A 60 GOPS/W, -1.8 V to 0.9 V body bias ULP cluster in 28 nm UTBB FD-SOI technology," *Solid-State Electronics*, vol. 117, pp. 170 – 184, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038110115003342

[15] D. Rossi *et al.*, "193 MOPS/mW @ 162 MOPS, 0.32V to 1.15V Voltage Range Multi-Core Accelerator for Energy-Efficient Parallel and Sequential Digital Processing," in *Cool Chips*, 2016.

[16] F. Conti and L. Benini, "A Ultra-low-energy Convolution Engine for Fast Brain-inspired Vision in Multicore Clusters," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15. San Jose, CA, USA: EDA Consortium, 2015, pp. 683–688.

[17] M. Gautschi *et al.*, "Tailoring instruction-set extensions for an ultra-low power tightly-coupled cluster of openrisc cores," in *VLSI-SoC 2015*, Oct 2015, pp. 25–30.

[18] A. Teman, D. Rossi, P. Meinerzhagen, L. Benini, and A. Burg, "Power, area, and performance optimization of standard cell memory arrays through controlled placement," in *ACM TODAES*, 2016.

[19] E. Beigne *et al.*, "A fine grain variation-aware dynamic vdd-hopping avfs architecture on a 32nm gals mpsoc," in *ESSCIRC 2013*, Sept 2013, pp. 57–60.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *arXiv:1409.4842 [cs]*, Sep. 2014.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[22] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both Weights and Connections for Efficient Neural Network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[23] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini, "Enabling the Heterogeneous Accelerator Model on Ultra-Low Power Microcontroller Platforms," in *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '16. San Jose, CA, USA: EDA Consortium, 2016.