PULP: A Ultra-Low Power Parallel Accelerator for Energy-Efficient and Flexible Embedded Vision

Article *in* Journal of Signal Processing Systems · September 2016

DOI: 10.1007/s11265-015-1070-9

5 authors, including:

Francesco Conti
ETH Zurich
67 PUBLICATIONS **943** CITATIONS
SEE PROFILE

Davide Rossi
University of Bologna
142 PUBLICATIONS **2,267** CITATIONS
SEE PROFILE

Antonio Pullini
ETH Zurich
60 PUBLICATIONS **1,719** CITATIONS
SEE PROFILE

Luca Benini
University of Bologna
1,161 PUBLICATIONS **38,652** CITATIONS
SEE PROFILE

PULP: A Ultra-Low Power Parallel Accelerator for Energy-Efficient and Flexible Embedded Vision

Francesco Conti, Davide Rossi, Antonio Pullini, Igor Loi, Luca Benini

Version accepted for publication in the Springer Journal of Signal Processing Systems
Published version available online: http://link.springer.com/article/10.1007/s11265-015-1070-9
DOI: 10.1007/s11265-015-1070-9

Abstract Novel pervasive devices such as smart surveillance cameras and autonomous micro-UAVs could greatly benefit from the availability of a computing device supporting embedded computer vision at a very low power budget. To this end, we propose PULP (Parallel processing Ultra-Low Power platform), an architecture built on clusters of tightly-coupled OpenRISC ISA cores, with advanced techniques for fast performance and energy scalability that exploit the capabilities of the STMicroelectronics UTBB FD-SOI 28nm technology. We show that PULP performance can be scaled over a 1x-354x range, with a peak theoretical energy efficiency of 211 GOPS/W. We present performance results for several demanding kernels from the image processing and vision domain, with post-layout power modeling: a motion detection application that can run at an efficiency up to 192 GOPS/W (90% of the theoretical peak); a ConvNet-based detector for smart surveillance that can be switched between 0.7 and 27fps operating modes, scaling energy consumption per frame between 1.2 and 12mJ on a 320x240 image; and FAST+Lucas-Kanade optical flow on a 128x128 image at the ultra-low energy budget of 14 µJ per frame at 60fps.

This work was funded by the project IcySoC, evaluated by the Swiss NSF and funded by Nano-Tera.ch with Swiss Confederation financing, and by the EU FP7 project PHIDIAS (g.a. 318013). We also thank STMicroelectronics for granting access to the FDSOI 28nm technology libraries.

Francesco Conti · Davide Rossi · Igor Loi · Luca Benini
Department of Electrical, Electronic and Information Engineering, University of Bologna
E-mail: {f.conti,davide.rossi,igor.loi,luca.benini}@unibo.it

Antonio Pullini · Luca Benini
Integrated Systems Laboratory, ETH Zurich,
E-mail: {pullinia,lbenini}@iis.ee.ethz.ch

Keywords Ultra-Low Power · Embedded vision · Convolutional Neural Network · Optical Flow · Motion Estimation · FD-SOI · Multi-core · OpenRISC

1 Introduction

With the introduction of cheap and powerful embedded computing devices such as Qualcomm Snapdragon 810 [**?**] and Nvidia Tegra K1 [**?**], the computer vision field has started to shift from theory and PC-based prototypes towards embedded applications such as smart cameras, self-driving cars and semi-autonomous robots. However, all current vision devices depend on the availability of a relatively abundant source of energy such as a mobile phone battery, which prevents integration of significant vision capabilities in devices that must run on very limited power and energy budgets, such as micro- or nano-UAVs that have a limited payload to host a battery or wireless sensor nodes (WSNs) that run on harvested power or must live years on a single charge [**?**]. These devices typically employ low power and ultra-low power microcontroller units (MCUs) that cannot cope with the heavy workloads of CV algorithms, even for very small images.

The ideal computing platform for this kind of heavily energy-constrained applications would be a low power, yet flexible fabric that is able to provide significant performance when needed and remain in a very low-consumption state when not. In particular, smart cameras, micro-UAVs and other similarly constrained applications that are designed to work with input from low-power imagers and performing vision-related algorithms need an exceptional degree of performance and energy scalability to cope both with the limited energy

budget and with the frame-rate requirements of vision applications. At the same time, a computing fabric answering to these needs should also provide a very high level of programmability with an easy-to-use model, to keep on track with the fast-moving CV field.

In this work we introduce *PULP* (Parallel processing Ultra-Low Power platform), a many-core platform answering to these demands. To achieve high performance when needed, PULP features clusters of simple, yet complete, OpenRISC [**?**] cores that can be used to exploit both coarse- and fine-grain data level parallelism or task level parallelism. At the same time, operating points (voltage, frequency, body biasing) can be controlled at a fine granularity and high speed to achieve high energy efficiency when the performance constraints are more relaxed or when the power budget is tighter. The proposed PULP platform exploits the capabilities of STMicroelectronics Ultra-Thin Body and Box Fully Depleted Silicon-on-Insulator (UTBB FD-SOI) technology [**?**] that, in contrast with deep submicron bulk technologies, allows to exploit an extended body bias range to modulate the performance/energy trade-off at different operating points.

We put our platform to test using several vision benchmarks, which were implemented in pure C code using the OpenMP programming model to express parallelism. Two benchmarks are targeted at the smart surveillance use case. The first is absolute difference motion estimation, a well known highly parallel algorithm that can be used to detect intruders in a camera stream, and is also a component of successful video compression algorithms[**?**]. The second benchmark is based on Convolutional Neural Networks (*CNN*s or *ConvNet*s) [**?**], a model that is state-of-art in many current CV benchmarks and has shown promising accuracy results in new classification, detection, and full-scene understanding tasks. CNN-based algorithms are typically computationally demanding and require a good level of performance to work at acceptable frame rates. Finally, to demonstrate the micro-UAV use case for a device such as PULP, we provide a benchmark based on Lucas-Kanade optical flow [**?**] that can be used as input for self-stabilization and hovering in an aerial vehicle.

The paper is organized as follows: Section 2 reports related works, introducing the state of art of research on energy-efficient embedded computing platforms, particularly those devoted to embedded vision. Section 3 overviews the architecture of the PULP platform, and its features to support low-power computing. Section **??** details our implementation results, it compares PULP with several other platforms and analyzes performance and energy efficiency in three benchmarks: motion detection, convolutional neural networks and optical flow.

2 Related work

Architectural research on many-core architectures has focused on tiled platforms; each tile contains one or more cores and communicates with other tiles through a scalable medium. The dominating paradigm is that of general-purpose and embedded GPUs such as NVIDIA Tegra [**?**]. GPUs feature a restricted SPMD-based execution model that can be suboptimal for CV applications, which have often an irregular structure [**?**][**?**]. Many-core platforms with clusters of RISC cores have been proposed as a more flexible model: examples include STMicroelectronics *P2012* [**?**], which is programmable in OpenCL [**?**] and OpenMP [**?**]; and Kalray *MPPA* [**?**], which supports a proprietary KPN-based programming model as well as OpenMP. All these platforms target a different power budget (from a few watts to several hundred milliwatts) with respect to the PULP platform we present in this paper.

To improve energy efficiency, many CV-targeted platforms rely on clusters of VLIW cores; for example, Movidius *Myriad* [**?**] features 8 SHAVE clusters, each including a VLIW core. Other examples are the TI *AccelerationPAC* [**?**], which includes several EVE clusters composed of a RISC processor and a VLIW coprocessor, and the Qualcomm Hexagon DSP [**?**] that accelerates a Snapdragon 800 with VLIW DSPs. While it is technically possible to develop new functionality for these platforms, all heavily rely on intrinsics, specialized assembly languages and other very low level programming models to deliver maximum performance and efficiency.

Trading off flexibility for additional efficiency, many CV-focused platforms rely on fixed-function HW blocks. Most of these platforms are dataflow engines, often implemented on FPGAs or CGRAs. Examples of this approach include *Vortex* [**?**][**?**] for biologically-inspired vision acceleration, and *NeuFlow* [**?**] and TeraDeep *nn-X* [**?**], which focus on ConvNet acceleration. Another high-performance platform based on a combination of coarse- and mid-grain reconfigurable blocks and embedded FPGA blocks is *Morpheus* [**?**], that can reach up to 50 GOPS/W energy efficiency on a variety of applications. Also some commercial products follow this path: for example the Analog Devices *Blackfin* [**?**] features a fixed-function Pipelined Vision Processor for CV acceleration. Another approach is to augment an existing many-core with fixed-function accelerators or coprocessors, as is done in *He-P2012* [**?**].

None of the platforms reported above currently targets ultra-low power operation, as their power budget ranges from hundreds of milliwatts to several watts. At the other end of the spectrum, microcontrollers can eas-

ily target power budgets of 50 mW and below, even in the case of high performance MCUs such as the STMicroelectronics STM32F401, based on a ARM Cortex-M4 [?]. State-of-art ULP microcontrollers can work with less than 10 mW: examples include the SiliconLabs *EFM32* [?], Texas Instruments *MSP430* [?] families of MCUs, and *Ambiq Apollo* [?]. Significant efficiency can be reached by near-threshold microcontrollers such as the one shown in Ickes et al. [?], *SleepWalker* [?] and *Bellevue* [?], which also exploits SIMD parallelism to further improve performance.

Since the performance level attainable by these low-power MCUs is too low for most CV applications, many CV-focused ULP accelerators employ fixed-function HW blocks [?][?][?]. Application specific architectures for low-power computer vision include designs trying to exploit alternative approaches to computation with respect to the traditional Von-Neumann ones; examples include state-of-art ASICs implementing spiking neural networks [?][?]. In some cases computation may be performed in the analog domain, for example in mixed-signal ASICs based on cellular neural networks [?]; another example is that of integrating simple filtering capabilities directly within the vision sensor [?].

A class of designs that are more directly comparable to our PULP platform is that of parallel low-power processors. *Centip3de* [?] consists of a large scale 3D-integrated fabric of clusters of Cortex M3 cores. With 64 cores running at 10 MHz, it can reach a performance of 0.64 GOPS. *DietSODA* [?] features 128 SIMD lanes working at lower frequency (50 MHz) than the rest of the chip, reaching up to 6.4 GOPS. These multicore platforms achieve efficiency levels comparable to those of PULP, but they compromise on programmability and flexibility; moreover, to the extent of our knowledge results on their power/performance scalability have not been provided. Dogan et al. [?] explore multicore design in subthreshold for biomedical usage, with a power budget as low as 10 µW that is more than one order of magnitude the power budget of PULP; however, this comes at a huge cost in terms of performance that also hits overall energy efficiency. To the best of our knowledge, the only commercial ULP multicore microcontroller on the market is the NXP LPC51400 [?], that asymmetrically couples a Cortex-M4 powerful microcontroller with a low-power Cortex-M0 for sensor control.

The first application we chose to evaluate PULP is smart visual surveillance, with the motion estimation and CNN benchmarks. Motion estimation is a well-known algorithm that is part of video standards such as MPEG [?], with known hardware (e.g. Hsieh et al. [?]) and software (e.g. Brockmeyer[?]) implementations. Conversely, Convolutional Neural Networks (CNNs or

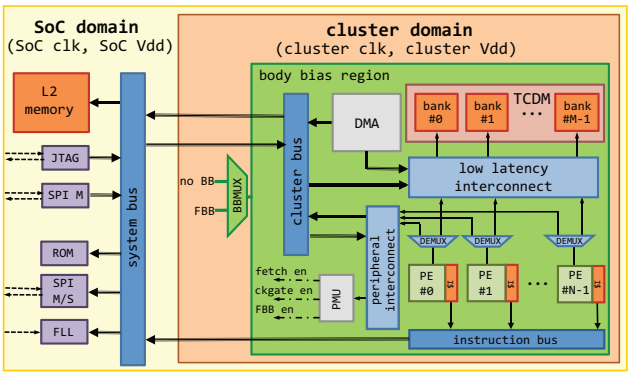


Fig. 1: PULP architecture.

ConvNets), originally proposed by LeCun et al. [?], have been object of many recent developments that were rekindled by the discovery of efficient ways to train them [?]. ConvNets have been used to obtain state-of-art accuracy results on scene labeling, video classification and object detection and interest in their applications has been shown by companies such as Google [?][?], Microsoft [?] and Facebook [?].

Future applications for scalable ultra-low power and energy computing devices are beginning to emerge in many fields, such as that of micro-UAVs and of smart and ubiquitous surveillance. State-of-art work on autonomous UAVs focuses on relatively big UAVs that are driven by full desktop-class processors and GPUs [?] [?]; to achieve full autonomy in micro-UAVs with much more limited batteries and payload a breakthrough in computing efficiency is needed. Wood et al. [?] quantify the total power budget for this kind of vehicle as 102 mW, of which only 5 mW can be dedicated to sensing and computation. In a similar fashion, smart wireless cameras acting as wireless sensor network nodes need to perform relatively complex activities in a reduced amount of time, while keeping the energy consumption at a minimum [?].

3 Architecture

3.1 PULP SoC overview

PULP (Parallel processing Ultra Low Power platform) is a scalable, clustered many-core computing platform able to operate on a large range of operating voltages, achieving in this way a high level of energy efficiency over a wide range of application workloads. Figure 1 shows the main building blocks of a single-cluster SoC. The PULP fabric is integrated in a SoC featuring a L2 memory (sized in the 32 kB to 128 kB range) shared among all

clusters through a system bus, plus IO peripherals that provide flexibility to the whole platform.

The set of peripherals integrated in the PULP platform includes two SPI (Serial Peripheral Interface) interfaces (one master and one slave), GPIOs, a bootup ROM and a JTAG interface suitable for testing purposes. Both SPI interfaces can be configured in *single* mode or *quad* mode depending on the required bandwidth, and they are suitable for interfacing the SoC with a large set of off-chip components (non volatile memories, voltage regulators, cameras...). Moreover, the SPI slave can be configured as a master, and a set of enable signals placed on both SPI interfaces allow the SoC to interface to up to 4 slave peripherals.

Thanks to its peripheral architecture the SoC is able to operate in two different modes: *slave* mode or *stand-alone* mode. When configured in slave mode, PULP behaves as a many-core accelerator of a standard *host* processor (e.g. an ARM Cortex M low-power microcontroller). In this configuration the host microcontroller is responsible for loading the application and processing data on the PULP L2 through the SPI master interface, and initiate and synchronize the computation through dedicated memory mapped signals (e.g. fetch enable) and GPIOs. When configured in stand-alone mode the SoC detects the presence of a flash memory on its SPI master interface, booting from the external flash if connected, from the L2 memory otherwise.

3.2 Cluster architecture

The cluster architecture features a parametric number of Processing Elements (*PE*s) consisting of a highly power optimized microarchitecture based on OpenRISC 32-bit ISA [**?**], each one with a private instruction cache (*I$*). The refill ports of all instruction caches converge on a common cluster instruction initiator port through a cluster instruction bus. The OpenRISC cores were optimized to achieve a high IPC on a wide variety of benchmarks, including control-intensive code [**?**]. Energy efficiency is boosted by using a flat pipeline to reduce register and clocking overhead, while the datapath was area-optimized to reduce leakage. Further, extensive architectural clock gating was employed to reduce spurious dynamic power.

The PEs do not have private data caches, avoiding memory coherency overhead and increasing area efficiency, while they all share a L1 multi-banked tightly coupled data memory (*TCDM*) acting as a shared data scratchpad memory. The TCDM has a number of ports equal to the number of memory banks providing concurrent access to different memory locations. Intra-cluster communication is based on a high bandwidth *low-latency interconnect*, implementing a word-level interleaving scheme to reduce access contention [**?**].

A lightweight, ultra-low-programming-latency, multi-channel DMA enables fast and flexible communication with other clusters, the L2 memory and external peripherals [**?**]. The DMA uses minimal request buffering and features a direct connection to the TCDM, to eliminate the need for internal buffering, which is very expensive in terms of power. A peripheral interconnect provides access to all the cluster peripherals and to all the resources external to the cluster.

3.3 Power management

In order to provide the best energy efficiency across a wide range of workloads, each cluster can work at its own voltage and frequency. To enable fine grained tuning of the SoC frequency, a FLL (Frequency-Locked Loop [**?**]) is included as a peripheral at SoC level. Moreover, a set of clock dividers (one for the SoC + one for each cluster) allow to further divide the clock generated by the FLL. To reduce the dynamic power consumption in idle mode, each processor can be separately disabled and clock-gated through a set of registers mapped on the peripheral interconnect. In this way, depending on the required workload, each cluster is able to work with an arbitrary number of processing elements, while the others consume zero dynamic power.

A body bias multiplexer (*BBMUX*) allows to dynamically select the back-bias voltage of the cluster, enabling ultra-fast transitions between the normal operating mode and the boost mode when temporary peaks of computation are required by the applications. To reduce the latency of the transitions between different operating modes, and making them transparent to the software, a power management unit (*PMU*) was added to generate the control signals of the processors fetch enables, clock gating units, and BBMUX.

4 Benchmarking PULP

This section examines the implementation results of the PULP platform on a reference configuration, providing an estimation of the area of the platform, of the energy efficiency at the different operating points, and a comparison with other state of the art multi-core platforms for embedded computing. It also reports the results of our evaluation of performance and energy in the motion estimation, ConvNet and optical flow benchmarks.

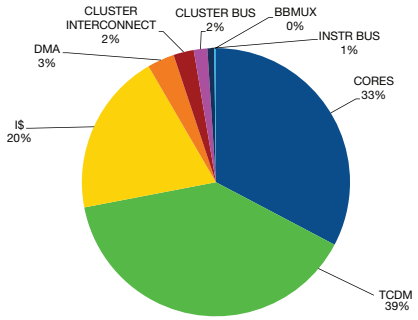


Fig. 2: PULP cluster area breakdown.

V_{DD} [V]	f_{max}[MHz] $V_{FBB} = 0V$	f_{max}[MHz] $V_{BB} = 0.5V$	f_{max}[MHz] $V_{BB} = 1V$
0.3	**2.5**	4.45	6.31
0.4	**22**	35.9	**49.1**
0.6	**200**	**277**	**350**
0.8	400	**484**	**563**
1.0	588	**650**	**705**
1.3	**775**	**836**	885

Table 1: Supply voltage and peak frequencies for the reference PULP cluster. Bold values indicate reference operating points.

4.1 Implementation results

In the context of this work we consider a single cluster PULP implementation operating in stand-alone mode. Thus, we assume the SoC connected to an external flash memory which contains the application code, a video surveillance camera periodically feeding the L2 of the SoC with a new frame, and a programmable DC/DC converter configured by the cores to switch between the idle, *search* and *follow* mode described in Section **??**. The L2 memory was sized at 128 kB to fit both the program code and one 320x240 frame. The cluster consists of 8 cores featuring 1 kB of I$ each, while the TCDM is composed of 16 banks of 2 kB each, leading to an overall TCDM size of 32 kB. These architectural parameters were chosen to fit the constraints of the benchmarks described in Section **??**, and should be sufficiently flexible for a broad variety of vision tasks. Both the TCDM banks and the processor's I$ are implemented using standard cell memory (SCM) cuts of 4 kbits each. While SRAMs may achieve a higher density than SCMs (by a factor of ∼3x), SCMs are able to work at the same voltage ranges as the rest of the logic, with the key benefit of providing much smaller energy/access (∼4x)[**?**].

Our results refer to a post place & route implementation of the proposed SoC in STMicroelectronics 28nm UTB FD-SOI technology. Thus, they include the overheads (i.e. timing, area, power) caused by the clock tree implementation, accurate parasitic models extraction, cell sizing for setup fixing and delay buffers for hold fixing (neglecting these would cause significant underestimations in the clock tree dynamic power). The SoC was synthesized with Synopsys Design Compiler, the place & route was performed using Cadence SoC Encounter, and the signoff was performed using Synopsys StarRC for parasitic extraction and Synopsys PrimeTime for timing and power analysis.

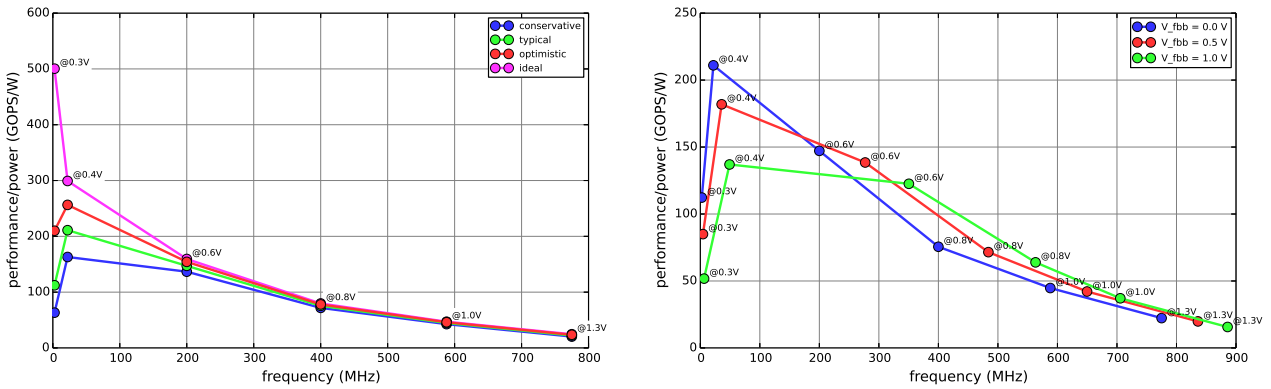
We tested our platform with power supplies ranging from 0.3V to 1.3V and forward body biasing ranging from 0 to 1V in the typical corner case at the temperature of 25°C. Table **??** shows the peak frequency that the PULP cluster can reach at each operating point. Being the cluster composed of 8 cores, the theoretical performance of the platform can easily scale between 20 MOPS @0.3V, no BB to 7 GOPS @ 1.3V, 1.0V FBB, demonstrating the dramatic performance scalability (354x) that can be exploited on PULP.

Figure **??** shows the area breakdown of the cluster, where the overall cluster area in the considered configuration is 1.2 mm^2. It is possible to note that the TCDM and the cores I$ occupy ∼59% of the overall cluster area, mainly due to the SCM based implementation. However, this is fully compensated by the improvement in terms of dynamic power consumption of the memories, which are responsible for the ∼15% of the overall cluster dynamic power, with an improvement of ∼4x with respect to a previous implementation of the same architecture [**?**].

4.2 Energy efficiency analysis

This section provides an evaluation of the energy efficiency of the proposed PULP implementation at the different operating points that can be exploited on the platform. To cope with the leakage power variation in the 28nm UTB FD-SOI, cell libraries are characterized very conservatively; early silicon measurements on PULP prototypes showed that there is more than a 2x guardband on power models. For this reason, we first evaluated the energy efficiency of the platform in four scenarios, accounting for various levels of pessimism for leakage: *conservative*, where the leakage power is directly extracted from the standard cell libraries; *typical*, with leakage scaled down by 2x; *optimistic*, where it is scaled down by 5x; and *ideal* with no leakage. This experiment allowed us to quantify the impact of the leakage power model guardband over our energy efficiency estimation.

Figure **??** shows the results of this exploration; the platform is working at the maximum operating frequency



(a) GOPS/W while scaling the leakage contribution to power.

(b) GOPS/W with 0V, 0.5V and 1.0V FBB.

Fig. 3: PULP energy efficiency in GOPS/W.

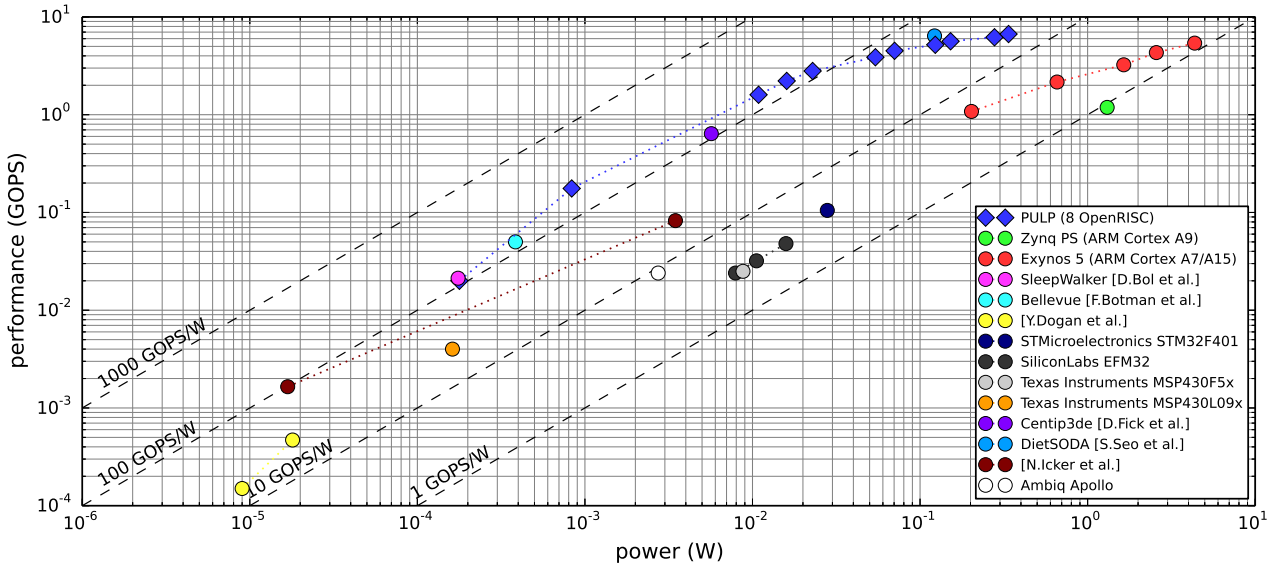


Fig. 4: Energy efficiency comparison with several platforms.

achievable at each given supply voltage. The peak energy efficiency points in the four scenarios are 172 GOPS/W, 211 GOPS/W, 262 GOPS/W, and 500 GOPS/W respectively. The best energy efficiency point is around 0.4V in all the scenarios except for the ideal. In all but the ideal scenario, the impact of leakage power is huge in the 0.3V to 0.4V operating range, when the supply voltage V_{DD} is close to V_{th} (0.28V for this technology), due to the relatively slow operating frequency (2.5 MHz to 50 MHz) that causes the static contribution of leakage to be dominant. On the other hand, when working with V_{DD} larger than 0.6V, the combined effect of increased dynamic power density (which scales as V_{DD}^2), and higher operating frequency causes the impact of leakage to be smaller. In the rest of the paper we only consider the typical scenario with a twofold leakage reduction as our reference for further power estimations and comparisons; measurements on a previous batch of fabricated PULP prototypes suggest that this is the most realistic value.

Figure **??** shows what happens when forward body biasing (FBB) is introduced. By applying FBB, it is possible to dynamically modulate the V_{th} of transistors to improve the frequency without changing the supply, with only a slight increase of dynamic power in the high-V_{DD} range. On the other hand, FBB introduces an overhead in leakage power, quantifiable as a 7x increase when V_{BB} is 1V [**?**]. For these reasons, FBB is an effective knob to increase the energy efficiency by up to 1.5x for workloads larger than 1.6 GOPS (200 MHz). For example, the target workload of 3.2 GOPS (400 MHz)

can be achieved @0.8V with 0V FBB or @0.6V with 1V FBB, resulting in a 1.5x improvement in energy efficiency.

To further provide insight into the scaling capabilities of the PULP platform, in Figure **??** we investigate energy efficiency in terms of peak GOPS per watt. We compare the reference PULP platform with several other commercial and academic platforms: the Processing System of the Xilinx Zynq platform (i.e. a dual core ARM Cortex A9), a Samsung Exynos 5 (i.e. a ARM big.LITTLE quad-core A7 + quad-core A15), and many of the ULP platforms referenced in Section 2. PULP, providing up to 211 GOPS/W, is competitive with microcontrollers specialized for low-power (Bellevue, SleepWalker) and more performant parallel ULP platforms (Centip3de, DietSoda), and is much more efficient than mobile solutions such as the Exynos 5 due to the simpler, optimized architecture of the OpenRISC cores and to the fine-grain knobs for power management provided by the FDSOI technology. It must also be noted that both Centip3de and DietSoda do not support a programming model, whereas PULP has been designed for compatibility with standards such as OpenCL and OpenMP, to ease the exploitation of potential performance in applications.

4.3 Motion estimation benchmark

As a first test for the PULP cluster, we wrote an absolute difference motion estimation [**?**] benchmark composed of several simple kernels: background subtraction, absolute value, binarization, erosion, dilation and a Sobel filter. The aim of the proposed algorithm is to detect the presence of external objects on a video transmitted by a camera framing a fixed background. For each video frame the first stage performs the absolute difference between the current and the background image. The resulting maximum value is extracted and used to calculate the threshold for binarization. The binarized image is then processed by three spatial operators. Erosion and dilatation implement the opening kernel which denoises the binarized image, while edge detection is implemented through a bidimensional Sobel convolution filter to create the external object boundary. If an external object is detected, the final kernel returns the highlighting of that object on the original frame.

The motion estimation benchmark runs on an input 8-bit grayscale 176x120 QCIF image produced by a low-power camera and loaded on the PULP L2 memory along with a prerecorded background. The program code occupies 12508 bytes in the L2 memory. Since the full image cannot fit in the TCDM, we divided the input image in slices or *tiles* of 44x20 pixels that are loaded into the TCDM and processed separately. Each tile occupies 1320 bytes in the TCDM, with a total TCDM occupation of 10560 bytes (four buffers used for present tile output computation, plus four used for double buffering).

Figure **??** shows the speedup of parallel versus sequential execution. This kernel is relatively simple and linear and completely parallelizable; as a consequence, its performance scales nicely up to 16 cores. The slight gap between the theoretical and simulated performance is mainly caused by the calculation of the maximum pixel value after the binarization stage, that cannot be completely parallelized over the available cores. Even so, as that is the only sequential part of the benchmark, speedup and energy efficiency are almost ideal. Figure **??** shows that energy efficiency of the motion estimation benchmark peaks at 192 GOPS/W at the 0.4V operating point, 90% of the theoretical limit.

4.4 ConvNet benchmarks

4.4.1 Convolution-accumulation optimization

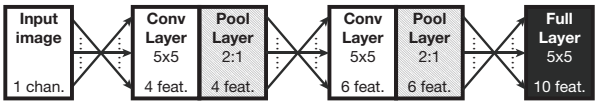
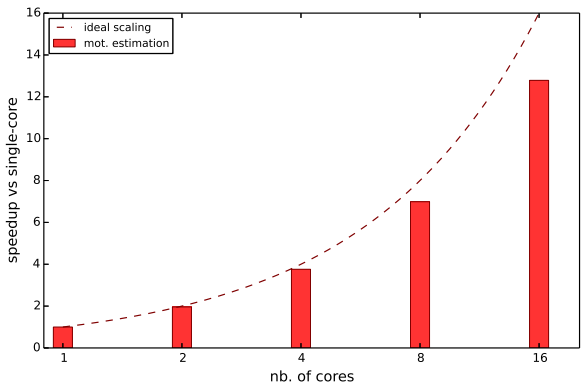


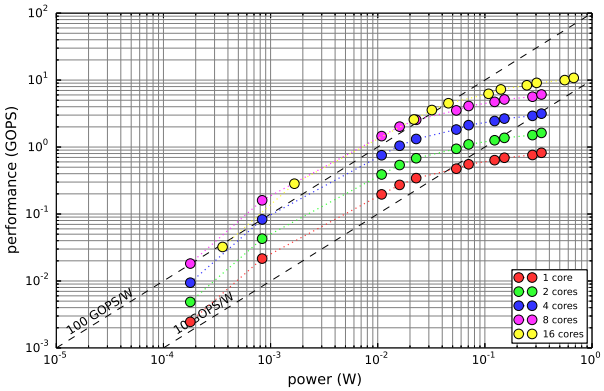
Fig. 6: Reference convolutional network.

A CNN is composed by a deep sequence of convolutional or fully-connected linear layers intermixed with pooling ones to perform a transformation on *feature maps* produced by the previous layer. Weights in convolutional and linear layers are trained by backpropagation but are used thereafter in a strictly feedforward fashion; due to their data parallel nature they are a natural candidate for acceleration in a parallel platform such as PULP. Convolutional layers in CNNs compute output feature maps of a layer as sums of convolutions over input feature maps; therefore, we chose to use a *convolution-accumulation* step as our basic kernel: $y(i,j) := y(i,j) + \Big(W * x\Big)(i,j)$. where x is the input image, W is the convolution kernel and y is the output image.

We used 16-bit fixed point numbers for inputs, kernels and outputs. We implemented three versions of convolution-accumulation: `naive` directly implements it as four nested loops (two on the output pixels and two for the convolution kernel W); `1-unrolled` uses manual loop unrolling on the innermost loop; `2-unrolled` uses loop unrolling on the two innermost loops. We bench-



(a) Parparallel speedup over sequential execution (1 tile).



(b) Energy efficiency (1 tile).

Fig. 5: Motion estimation benchmark results.

Implementation	**3x3**	**5x5**	**7x7**	**9x9**	**11x11**
`naive`, single thread	0.26	0.32	0.34	0.35	0.36
`1-unrolled`, single thread	0.52	0.62	0.65	**0.69**	**0.88**
`2-unrolled`, single thread	**0.80**	**0.83**	**0.76**	0.26	0.18
`naive`, 8 threads	0.26	0.31	0.34	0.35	0.36
`1-unrolled`, 8 threads	0.49	0.60	0.65	**0.69**	**0.85**
`2-unrolled`, 8 threads	**0.71**	**0.77**	**0.74**	0.27	0.18

Table 2: Convolution-Accumulation: average efficiency/-core

marked these convolutions with a single thread or 8 parallel threads[1].

Table **??** shows the efficiency/core for the various convolution-accumulation implementations on a 32x32 input image, computed as the ratio between useful (i.e. computation) cycles and the total number of cycles spent in the outermost loop. For smaller convolution kernels, unrolling both inner loops provides a much better efficiency; however, for kernels bigger than 7x7, efficiency is reduced by I$ misses due to the size of the unrolled loop. As a consequence, the tighter `1-unrolled` convolution-accumulation step is more convenient for bigger kernels. Results are similar in the multi-threaded case, as data contention on the TCDM causes on average only a small amount of efficiency decrease.

4.4.2 Use case: CNN for visual surveillance

On top of these optimized convolutions, we developed a network based on the one proposed by LeCun et al. [**?**] for MNIST classification, which is shown in Figure **??**. This network has 2220 parameters and a footprint of 11408 bytes for data and 4400 bytes for weights on the L1 TCDM; Table **??** summarizes them. The program code uses 16768 bytes on the L2 memory. As shown in Conti et al. [**?**], a network of this kind can be trained for complex object detection tasks by running it on a window sliding over the input frame.

We use this CNN for visual surveillance. The platform spends most of the time in a low-power *search* mode looking for suspicious objects (as this task requires only a relatively low frame rate), and it switches to a high-performance *follow* mode to keep track of a previously detected object. Input frames are brought inside the PULP cluster by DMA transfer from the L2. This transfer is superimposed to the computation of deeper layers and has no impact on the final throughput.

Figure **??** shows the performance of the reference CNN when run on a 32x32 image patch, scaling the clock frequency of the cluster from 100 MHz to 1 GHz and the number of OpenRISC cores in the PULP cluster between 1, 2, 4, 8 or 16. As expected from a highly data-parallel algorithm such as ConvNets, execution time scales almost linearly with the number of cores. In our visual surveillance application, the ConvNet is run on a 32x32 window spanning a QVGA (320x240) image with a stride of 32 pixels. Each frame is spanned two times: one with no offset, the other with an offset of 16 pixels in both directions so that the chance of missed detections on the border of a window are reduced. PULP can be set to work at a very low frame rate (∼0.7 fps at the 0.4V operating point) in the *search* mode, and then switched to a frame rate as high as 27 fps (at the 1.3V operating point with 1V FBB) in the *follow* mode.

Figure **??** shows the energy efficiency of the ConvNet execution on a frame in terms of FPS/W; we ran

[1] We used the `or1k-elf-gcc` compiler (build `4.9.0 20140308`), with the following flags: `-O2 -nostdlib -mhard-mul -msoft-div`.

layer	# feat.	params filter size	data size	memory (bytes) weights	data
input	1	-	32x32	0	2048
conv 0	4	5x5	28x28	200	6272
pool 1	4	-	14x14	0	1568
conv 2	6	5x5	10x10	1200	1200
pool 3	6	-	5x5	0	300
full 4	10	5x5	1x1	3000	20

(a) *small* CNN.

layer	# feat.	params filter size	data size	memory (bytes) weights	data
input	1	-	32x32	0	2048
conv 0	8	5x5	28x28	400	12544
pool 1	8	-	14x14	0	3136
conv 2	12	5x5	10x10	4800	2400
pool 3	12	-	5x5	0	600
full 4	10	5x5	1x1	6000	20

(b) *medium* CNN.

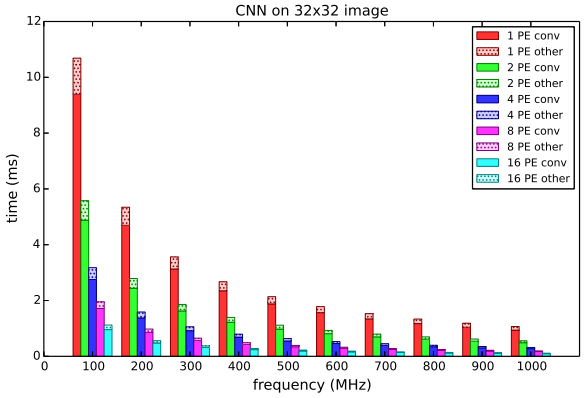
layer	# feat.	params filter size	data size	memory (bytes) weights	data
input	1	-	32x32	0	2048
conv 0	16	5x5	28x28	800	25088
pool 1	16	-	14x14	0	6272
conv 2	24	5x5	10x10	19200	4800
pool 3	24	-	5x5	0	1200
full 4	10	5x5	1x1	12000	20

(c) *big* CNN.

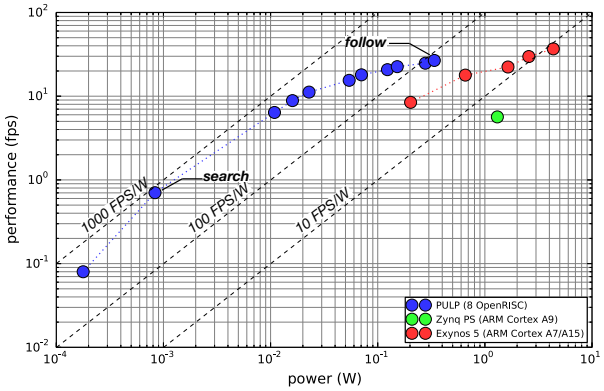
layer	# feat.	params filter size	data size	memory (bytes) weights	data
input	1	-	64x64	0	16384
conv 0	4	5x5	60x60	200	28800
pool 1	4	-	30x30	0	7200
conv 2	6	5x5	26x26	1200	8112
pool 3	6	-	13x13	0	2028
conv 4	10	5x5	9x9	3000	920

(d) *small* CNN on a 64x64 image.

Table 3: Parameters and memory usage of all CNN networks.



(a) Execution time in ms (32x32 patch).



(b) Energy efficiency in FPS/W (QVGA frame).

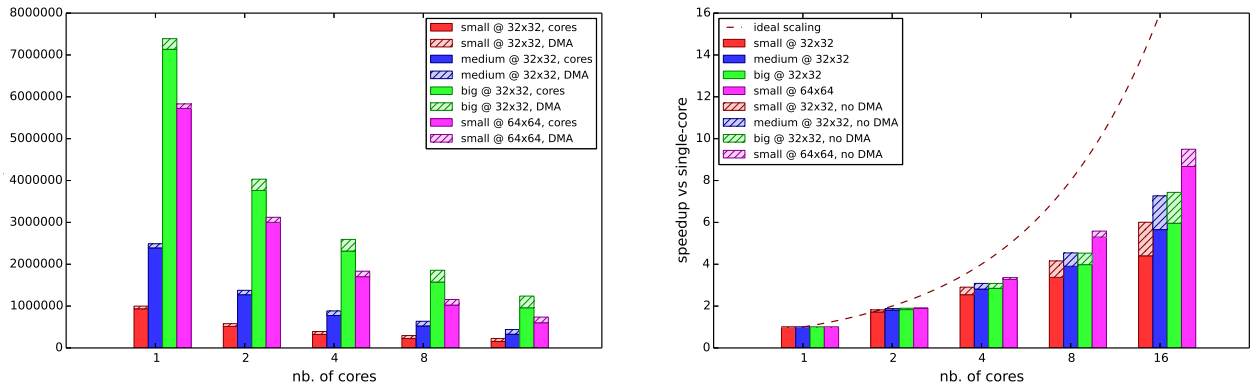
Fig. 7: Surveillance ConvNet benchmark results.

the same ConvNet on the Xilinx Zynq PS and on a Samsung Exynos 5 for comparison, as this benchmark is beyond the typical performance capabilities of most ULP microcontroller architectures. Benchmark results substantially confirm the theoretical values shown in Figure **??**. The energy/execution time tradeoff when switching between *search* and *follow* mode is also clearly shown: in *search* mode, PULP consumes 1.18 mJ per frame and lives at a power budget of 834 µW, whereas in *follow* mode energy consumption jumps at 12.6 mJ per frame.

4.4.3 Tiled CNNs

To further explore the capabilities of the PULP platform in this scenario, we considered the case that the CNN or its input image cannot fit in the TCDM. In this case, it is necessary to tile the CNN similarly to what is described in Section **??**; also in this case, double buffering can be employed to hide the latency of the L2/L1 memory transfer.

In the case of CNNs, tiling involves some amount of recomputation as the receptive field of each output convolutional tile is partially superimposed to that of



(a) Execution time in cycles.

(b) Parallel execution speedup.

Fig. 8: Tiled CNN benchmark performance results.

the next output tile. We can tile the same ConvNet with two distinct approaches. With a "vertical" tiling approach, the full network is applied to each input tile until the last layer, then the output is transferred to the L2 memory and a new tile is loaded; "horizontal" tiling instead is applied by dividing input of a single layer in tiles and computing all output tiles before proceeding to the following layer. In this approach, intermediate results (i.e. the outputs of intermediate layers) have to be stored in buffers in the L2 memory.

We chose to concentrate on horizontal tiling for three reasons: first, vertical tiling involves a lot of recomputation as the smaller ConvNet tile has to be moved over the input image (similarly to what we did in Section **??**, but with stride of 1 pixel instead of 32). Second, the horizontal approach allows us to tile also in the input feature dimension, whereas in vertical tiling all input features are needed in the shared memory to compute the following layer. Third, although horizontal tiling involves frequent data transfers between L1 and L2, we will show in the following that the impact of these transfers scales nicely with the size of the input data and the amount of parallelism.

We extended the reference CNN[2] of Section **??** in the following way. The *medium* and *big* CNNs, whose parameters are reported in Tables **??** and **??** respectively, are similar to the *small* one but their intermediate layers have more features. The fourth CNN shares the same parameters as the *small* one, but runs on a bigger image patch of 64x64 pixels; its parameters and memory consumption are reported in Table **??**. In this network, the final linear layer is substituted by an equivalent convolutional layer using the same weights; the output is equivalent to the separate classification of all pixels (see for example Sermanet et al. [**?**]). In all benchmarks, we set the maximum dimension of the tiles to 4KB so that it is possible to fit two input tiles and two output tiles in the TCDM. The dimension of the program code is similar for all of these benchmarks (∼25KB loaded on the L2 memory), since we relied on the same ConvNet library extended with horizontal tiling support.

Figure **??** reports the execution time of all benchmarks in terms of cluster clock cycles. The computational complexity of the CNN raises exponentially when we double the number of feature maps used in each layer or the pitch of the input image; we observe that the *big* CNN applied on a 32x32 image and the *small* one on a 64x64 image impose similar constraints both in terms of workload and of memory occupation. To better evaluate how performance scales in all benchmarks as we vary the number of cores, Figure **??** compares speedup versus single-core execution for all benchmarks. Figure **??** also reports the theoretical speedup if we neglected all impact of DMA transfers. The main limiting factor for speedup is given by Amdahl's law: due to the small dimension of the tiles, the parallel fraction of the code is not sufficient to yield quasilinear speedup. This is clearly visible in that the same ConvNet applied to a 4x bigger input image yields much better results in terms of performance scaling. The size of the input image itself is mainly limited by the availability of L2 memory. The plot also shows that, due to the higher computation to communication ratio, the impact of data transfers on the speedup scales nicely with the size of the workload, i.e. the bigger the input image and/or the CNN is, the less limiting impact DMA transfers have over parallel execution speedup.

[2] We will refer to this network as the *small* network from this point on.

To estimate how much the accuracy may vary between the *small*, *medium* and *big* CNNs, we trained them to classify the CIFAR-10 dataset [**?**], a well known and freely available set of 60000 32x32 images labeled in 10 classes [3]. Figure **??** shows that the difference can be significant: after 500 epochs of training the final accuracy is 70.64% for the *big* CNN, which drops to *64.38%* for the *medium* one and to 50.05% for the *small* one. The difference is greatly reduced if we compare the CNNs for a one versus all classification task over the same dataset: the final accuracy in this case is 95.1%, 94.2% and 93.3% for the *big*, *medium* and *small* CNNs respectively.

In Figure **??**, we plot the energy efficiency in terms of GOPS/W for the execution of the *big* ConvNet (results are practically identical for the other benchmarks). Compared with the peak theoretical value of 211 GOPS/W, we measured a peak of 150 GOPS/W in this benchmark, which correspond to an average IPC of 0.71 per core. By comparison, average single-core IPC in the inner convolutional loops is 0.96, and average single-core overall IPC is 0.87. The IPC reduction in the multi-core tests is mainly accounted for by contention on the shared TCDM and, to a lesser extent, by contention on the I$ refill bus. Still, IPC in the inner-loops is as high as 0.90 per core when executing with 8 cores. The energy efficiency results mimic the peak ones presented in Figure **??**, and peak efficiency (125 MOPS @ 834 µW) at the same operating point (0.4V without FBB) in the near-threshold region.

[3] As our CNNs work on grayscale images, the training and test samples where converted from RGB to grayscale. Training consisted in 500 epochs of mini-batch stochastic gradient descent with momentum $\mu = 0.9$ and starting learning rate $\lambda_0 = 0.01$ (dropping exponentially as $\lambda = \lambda_0 \cdot 0.995^{n_{\text{epoch}}}$), using 20% dropout [**?**] layers for better regularization.

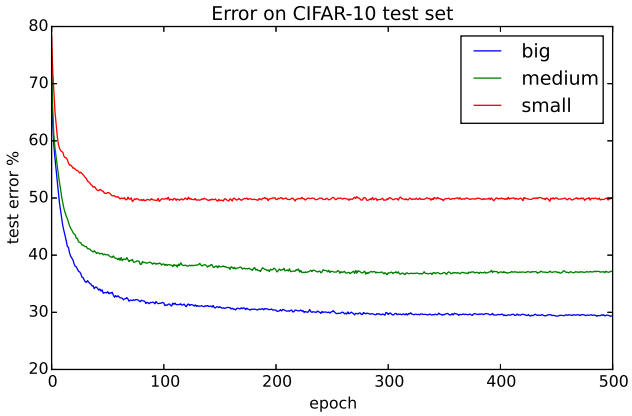


Fig. 9: Test error of *small*, *medium* and *big* CNNs on the CIFAR-10 set over 500 training epochs.

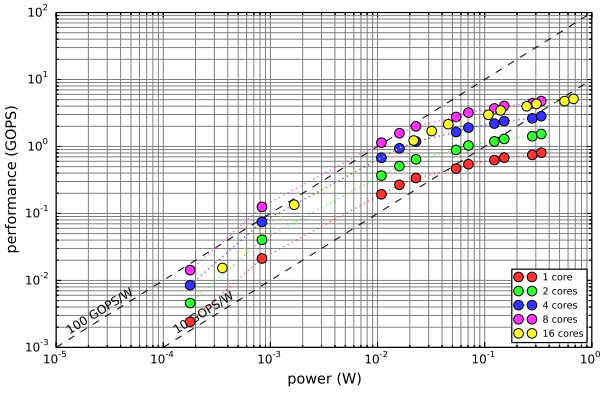


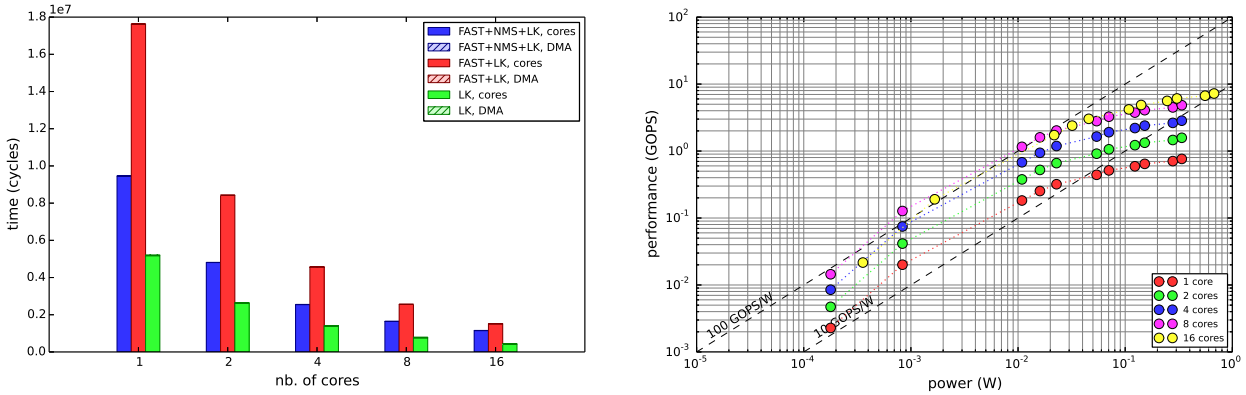
Fig. 10: Energy efficiency for execution of the *small* CNN on a 64x64 image, while sweeping the number of cores.

4.5 Optical flow benchmark

As a representative application for the usage of PULP as an accelerator for an autonomous nano-UAV, we developed an optical flow benchmark that is meant to be integrated in the drone control loop to make completely autonomous hovering and navigation possible. In this scenario, a low-resolution (e.g. 128x128 pixel) ultra-low-power imager such as a CentEye Stonyman [**?**] continuously feeds frames to PULP via the QSPI slave interface. On turn, PULP computes the optical flow and uses its QSPI master to report the flow vectors back to the microcontroller driving the vehicle, where they are used to estimate rotations and translations of the drone.

The benchmark is composed of three kernels: FAST corner detection [**?**][**?**], non-maximal suppression and Lucas-Kanade optical flow estimation [**?**]. Since Lucas-Kanade should be applied to strong corners to yield high-quality, it is generally not advisable to drop either the non-maximal suppression step or the whole corner detection. Nonetheless, since the users of the flow vectors (i.e. the aerial vehicle software developers) might want to trade off optical flow accuracy for performance and energy, we decided to explore also these non-optimal cases. Therefore, we present results for three separate implementations: *FAST+NMS+LK* that feeds corners produced by the FAST algorithm in non-maximal suppression before computing optical flow; *FAST+LK* that uses all corners produced by FAST for the optical flow; *LK* that drops corner detection and computes optical flow on all pixels.

The input of the optical flow application are two 128x128 8-bit grayscale frames stored in the L2 memory by the QSPI slave module. To cope with the dimension



(a) Execution time in cycles (all).

(b) Energy efficiency in GOPS/W (*FAST+NMS+LK*).

Fig. 11: Optical flow benchmark results.

of the input frames, we divided them in stripes of 128x16 pixels; we use double buffering to transfer the stripes from the L2 to the TCDM while we are computing the optical flow of the previous stripe.

Figure **??** reports the execution time in cycles for all versions of the benchmark, sweeping the number of cores in the PULP cluster from 1 to 16. The first observation is that the *FAST+LK* benchmark is the slowest; this is due to the fact that if non-maximal suppression is dropped, the Lucas-Kanade step has to be performed on a much higher number of corners, in the order of several hundreds. The *LK* benchmark drops FAST altogether and is therefore the fastest, even if it computes optical flow on the full 16384 pixels of the image. Conversely, the *FAST+NMS+LK* benchmarks spends most of its time in computing the best corners (in the order of some tens) in the picture and much less time in the actual optical flow, as it is computed only on those corners. In all cases, optical flow computation on the 128x128 input frames takes more than 1 million cycles when performed with 8 cores: intuitively, this means that the workload to perform this task at 60fps is bigger than 60 MOPS.

Figure **??** helps to understand whether this is a feasible target, and at what power budget, by plotting energy efficiency in terms of GOPS versus watt measured by profiling the optical flow *FAST+NMS+LK* benchmark. At the most efficient operating point (0.4V with no FBB, 834 µW of power consumption) the 8-core cluster achieves a performance of 127 MOPS, with an efficiency of 152 GOPS/W. The peak efficiency is similar to that of the CNN benchmark but this is due to a different mixture of effects from the result obtained in Section **??**. First, the lower internal regularity of the FAST benchmark (which is responsible for the majority of the execution time) hits the inner-loop IPC with respect to the very regular and manually optimized convolutional kernels employed in the ConvNet. At the same time, however, it also significantly lowers data contention, leading to a similar overall efficiency result. At this operating point, optical flow would be feasible for a micro-UAV application as it would add less than a mW to the total vehicle power, which nicely fits in the 5 mW budget for computing in Wood et al. [**?**]. The energy budget to compute a frame is 13.9 µJ; to make this measure concrete let us take for example the commercial Crazyflie Nano Quadcopter [**?**], that mounts a 240 mAh 3.3V battery, hosting approximately 2850 J of energy destined primarily to power DC motors. If we suppose a flight time of one hour, the battery consumption due to the PULP accelerator would amount to ∼3 J, i.e. 0.1% of the total battery, which is almost negligible with respect to the energy consumed by the vehicle actuators.

5 Conclusions

As our main contribution, we have introduced the PULP (*Parallel processing Ultra-Low Power*) platform that features clusters of tightly-coupled OpenRISC cores to achieve high energy efficiency through parallelism. We have analyzed the platform, showing that its performance can be scaled by the dramatic factor of 354x and that it features a peak energy efficiency of 211 GOPS/W.

As a use case for PULP, we show a motion estimation algorithm for smart surveillance which almost fully exploits the available performance, with a peak energy efficiency of 192 GOPS/W, i.e. 90% of the theoretical peak. We also implemented a ConvNet-based algorithm for video surveillance, showing that it can be switched from a low-power state consuming just 1.18 mJ per

frame with a rate of 0.7 fps to a high-performance state running at 27 fps and consuming 12.6 mJ per frame. Finally, we wrote a sample benchmark for applications in the nano-UAV field, where we use PULP to accelerate estimation of optical flow from frames produced by a ULP imager, with the objective of autonomous hovering and navigation; we show that it is possible to meet tight timing constraints (60fps frame rate) at the energy budget of 14 µJ per frame. These benchmarks showcase the high level of flexibility and programmability of the PULP platform, that does not come at the expense of energy efficiency: all of them were able to reach at least 70% of the peak efficiency overall, with much higher peaks in highly parallel regions such as in the motion detection and inner convolutional kernels.

A fully functional PULP test chip featuring 4 OpenRISC cores, 64 kB of L2 memory and 24 kB of TCDM has been submitted for fabrication in 28nm STMicroelectronics FD-SOI technology in December 2014. We decided to submit a 4-core version of the platform out of two main considerations: first, due to the relative novelty of the FD-SOI technology, we chose to be conservative and take into account a bigger leakage contribution to power consumption, which would level out the efficiency gap between 4- and 8-core PULP clusters. Second, the higher complexity of the shared-memory interconnect in a 8-core cluster with respect to a 4-core one coupled with the constrained wafer area allotted to our chips could make optimal placement & routing and timing closure more difficult, resulting in a slower or less energy-efficient chip.

Our work is now focusing on pushing the PULP architecture to the 1 GOPS/mW limit, making it competitive with special purpose mixed-signal accelerators such as the 1.57 TOPS/W in Kim et al. [**?**] in terms of energy efficiency, while also preserving general software programmability. This is being tackled with a complete redesign of the core, with microarchitectural changes to improve IPC and programmability, a new custom compiler toolchain based on LLVM and architectural improvements to the platform to improve its performance and energy efficiency, such as a more efficient instruction cache hierarchy and the addition of specialized blocks for selected macro-operations.

References

1. Ambiq Apollo website. URL `http://ambiqmicro.com/low-power-microcontroller`
2. Analog Devices Blackfin Dual Core Processor. URL `http://www.analog.com/en/processors-dsp/blackfin/adsp-bf608/products/product.html`
3. CentEye Stonyman / Hawksbill Silicon Documentation
4. CrazyFlie Nano-Quadcopter website. URL `http://www.bitcraze.se/crazyflie/`
5. NVidia Tegra K1 website. URL `http://www.nvidia.com/object/tegra-k1-processor.html`
6. NXP LPC54100 website. URL `http://www.nxp.com/products/microcontrollers/cortex_m4/lpc54100/`
7. Qualcomm Snapdragon 810 website. URL `https://www.qualcomm.com/products/snapdragon/processors/810`
8. SiliconLabs EFM32 Microcontroller. URL `http://www.silabs.com/products/mcu/lowpower/pages/efm32g-gecko.aspx`
9. STM32F401xB, STM32F401xC Data sheet
10. Texas Instruments MSP430 Low-Power MCUs. URL `http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/overview.page`
11. OpenRISC 1000 Architecture Manual (2012)
12. Benini, L., Flamand, E., Fuin, D., Melpignano, D.: P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator. In: 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 983–987. IEEE (2012). DOI 10.1109/DATE.2012.6176639. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6176639http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6176639`
13. Bol, D., De Vos, J., Hocquet, C., Botman, F., Durvaux, F., Boyd, S., Flandre, D., Legat, J.D.: SleepWalker: A 25-MHz 0.4-V Sub-mm2 7-uW/MHz Microcontroller in 65-nm LP/GP CMOS for Low-Carbon Wireless Sensor Nodes. IEEE Journal of Solid-State Circuits **48**(1), 20–32 (2013). DOI 10.1109/JSSC.2012.2218067. URL `http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6332542`
14. Botman, F., Vos, J.D., Bernard, S., Stas, F., Legat, J.D., Bol, D.: Bellevue : a 50MHz Variable-Width SIMD 32bit Microcontroller at 0 . 37V for Processing-Intensive Wireless Sensor Nodes. In: Proceedings of 2014 IEEE Symposium on Circuits and Systems, pp. 1207–1210 (2014)
15. Brockmeyer, E., Nachtergaele, L., Catthoor, F., Bormans, J., de Man, H.: Low power memory storage and transfer organization for the MPEG-4 full pel motion estimation on a multimedia processor. IEEE Transactions on Multimedia **1**(2), 202–216 (1999). DOI 10.1109/6046.766740
16. Carey, S.J., Lopich, A., Barr, D.R.W., Bin Wang, Dudek, P.: A 100,000 fps vision sensor with embedded 535gops/W 256x256 SIMD processor array. pp. C182–C183. IEEE, Kyoto (2013)
17. Chen, P., Hong, K., Naikal, N., Sastry, S.S., Tygar, D., Yan, P., Yang, A.Y., Chang, L.C., Lin, L., Wang, S., Lobatn, E., Oh, S., Ahammad, P.: A low-bandwidth camera sensor platform with applications in smart camera networks. ACM Trans. Sen. Netw. **9**(2), 21:1–21:23 (2013). DOI 10.1145/2422966.2422978. URL `http://doi.acm.org/10.1145/2422966.2422978`
18. Chua, L., Gulak, G., Pierzchala, E., Rodríguez-Vázquez, A.: Cellular neural networks and analog VLSI. Springer Science & Business Media (2013)
19. Clemons, J., Zhu, H., Savarese, S., Austin, T.: MEVBench: A mobile computer vision benchmarking suite. In: 2011 IEEE International Symposium on Workload Characterization (IISWC), pp. 91–102. IEEE (2011). DOI 10.1109/IISWC.2011.6114206. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6114206http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6114206`
20. Codrescu, L., Anderson, W., Venkumanhanti, S., Zeng, M., Plondke, E., Koob, C., Ingle, A., Tabony, C., Maule,

R.: Hexagon DSP: An Architecture Optimized for Mobile Multimedia and Communications. IEEE Micro **34**(2), 34–43 (2014). DOI 10.1109/MM.2014.12
21. Conti, F., Pilkington, C., Marongiu, A., Benini, L.: HeP2012 : Architectural Heterogeneity Exploration on a Scalable Many-Core Platform. In: Proceedings of 25th IEEE Conference on Application-Specific Architectures and Processors (2014)
22. Conti, F., Pullini, A., Benini, L.: Brain-inspired Classroom Occupancy Monitoring on a Low-Power Mobile Platform. In: CVPR 2014 Workshops (2014)
23. Deng, L., Li, J., Huang, J.T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., Gong, Y., Acero, A.: Recent advances in deep learning for speech research at Microsoft. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing pp. 8604–8608 (2013). DOI 10.1109/ICASSP.2013.6639345. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6639345`
24. de Dinechin, B.D., Ayrignac, R., Beaucamps, P.E., Couvert, P., Ganne, B., de Massas, P.G., Jacquet, F., Jones, S., Chaisemartin, N.M., Riss, F., Strudel, T.: A clustered manycore processor architecture for embedded and accelerated applications. In: 2013 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6. IEEE (2013). DOI 10.1109/HPEC.2013.6670342. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6670342`
25. Dogan, A.Y., Atienza, D., Burg, A., Loi, I., Benini, L.: Power/performance exploration of single-core and multi-core processor approaches for biomedical signal processing. In: Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation, pp. 102–111. Springer (2011)
26. Farabet, C., Martini, B., Corda, B., Akselrod, P., Culurciello, E., LeCun, Y.: NeuFlow: A runtime reconfigurable dataflow processor for vision. In: CVPR 2011 Workshops, pp. 109–116. IEEE (2011). DOI 10.1109/CVPRW.2011.5981829. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5981829`
27. Fick, D., Dreslinski, R.G., Giridhar, B., Kim, G., Seo, S., Fojtik, M., Satpathy, S., Lee, Y., Kim, D., Liu, N., Wieckowski, M., Chen, G., Mudge, T., Blaauw, D., Sylvester, D.: Centip3De: A Cluster-Based NTC Architecture With 64 ARM Cortex-M3 Cores in 3D Stacked 130 nm CMOS. IEEE Journal of Solid-State Circuits **48**(1), 104–117 (2013). DOI 10.1109/JSSC.2012.2222814. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6399548`
28. Gautschi, M., Rossi, D., Benini, L.: Customizing an open source processor to fit in an ultra-low power cluster with a shared L1 memory. In: Proceedings of the 24th edition of the great lakes symposium on VLSI - GLSVLSI '14, pp. 87–88. ACM Press, New York, New York, USA (2014). DOI 10.1145/2591513.2591569. URL `http://dl.acm.org/citation.cfm?doid=2591513.2591569`
29. Gokhale, V., Jin, J., Dundar, A., Martini, B., Culurciello, E.: A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks. In: CVPR 2014 Workshops
30. Heng, L., Honegger, D., Lee, G.H., Meier, L., Tanskanen, P., Fraundorfer, F., Pollefeys, M.: Autonomous visual mapping and exploration with a micro aerial vehicle. Journal of Field Robotics **31**(4), 654–675 (2014). DOI 10.1002/rob.21520. URL `http://onlinelibrary.wiley.com/doi/10.1002/rob.21520/abstract`
31. Hsieh, C.H., Lin, T.P.: VLSI architecture for block-matching motion estimation algorithm. IEEE Transactions on Circuits and Systems for Video Technology **2**(2), 169–175 (1992). DOI 10.1109/76.143416
32. Ickes, N., Sinangil, Y., Pappalardo, F., Guidetti, E., Chandrakasan, A.P.: A 10 pJ/cycle ultra-low-voltage 32-bit microprocessor system-on-chip. In: 2011 Proceedings of the ESSCIRC (ESSCIRC), pp. 159–162. IEEE (2011). DOI 10.1109/ESSCIRC.2011.6044889. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6044889`
33. Jacquet, D., Hasbani, F., Flatresse, P., Wilson, R., Arnaud, F., Cesana, G., Di Gilio, T., Lecocq, C., Roy, T., Chhabra, A., Grover, C., Minez, O., Uginet, J., Durieu, G., Adobati, C., Casalotto, D., Nyer, F., Menut, P., Cathelin, A., Vongsavady, I., Magarshack, P.: A 3 GHz Dual Core Processor ARM Cortex TM -A9 in 28 nm UTBB FD-SOI CMOS With Ultra-Wide Voltage Range and Energy Efficiency Optimization. IEEE Journal of Solid-State Circuits **49**(4), 812–826 (2014). DOI 10.1109/JSSC.2013.2295977
34. Jeon, D., Kim, Y., Lee, I., Zhang, Z., Blaauw, D., Sylvester, D.: A 470 mV 2.7mW Feature Extraction-Accelerator for Micro-Autonomous Vehicle Navigation in 28nm CMOS pp. 166–168 (2013)
35. Karpathy, A., Leung, T.: Large-scale Video Classification with Convolutional Neural Networks. In: Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732 (2014). DOI 10.1109/CVPR.2014.223
36. Kim, G., Kim, Y., Lee, K., Park, S.: A 1.22TOPS and 1.52mW/MHz Augmented Reality Multi-Core Processor with Neural Network NoC for HMD Applications. In: Proceedings of 2014 IEEE International Solid-State Circuits Conference, pp. 182–184 (2014)
37. Knag, P., Kim, J.K., Chen, T., Zhang, Z.: A Sparse Coding Neural Network ASIC With On-Chip Learning for Feature Extraction and Encoding. IEEE Journal of Solid-State Circuits **50**(4), 1070–1079 (2015). DOI 10.1109/JSSC.2014.2386892
38. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images (2009)
39. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)
40. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). DOI 10.1109/5.726791. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=726791`
41. Lin, Z., Sankaran, J., Flanagan, T.: Empowering automotive vision with TIs Vision AccelerationPac. TI White Paper (2013)
42. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81, pp. 674–679. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1981). URL `http://dl.acm.org/citation.cfm?id=1623264.1623280`
43. Ma, K.Y., Chirarattananon, P., Fuller, S.B., Wood, R.J.: Controlled Flight of a Biologically Inspired, Insect-Scale Robot. Science **340**(6132), 603–607 (2013). DOI 10.1126/science.1231806. PMID: 23641114
44. Mahesri, A., Johnson, D., Crago, N., Patel, S.J.: Tradeoffs in designing accelerator architectures for visual computing. In: 2008 41st IEEE/ACM International Symposium on Microarchitecture, pp. 164–175. IEEE (2008). DOI 10.1109/

MICRO.2008.4771788. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4771788
45. Marongiu, A., Capotondi, A., Tagliavini, G., Benini, L.: Improving the programmability of STHORM-based heterogeneous systems with offload-enabled OpenMP. In: Proceedings of the First International Workshop on Many-core Embedded Systems - MES '13, pp. 1–8. ACM Press, New York, New York, USA (2013). DOI 10.1145/2489068.2489069. URL http://dl.acm.org/citation.cfm?doid=2489068.2489069
46. Meinerzhagen, P., Sherazi, S.M.Y., Burg, A., Rodrigues, J.N.: Benchmarking of Standard-Cell Based Memories in the Sub-Vt Domain in 65-nm CMOS Technology. IEEE Journal on Emerging and Selected Topics in Circuits and Systems **1**(2), 173–182 (2011). DOI 10.1109/JETCAS.2011.2162159. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5976987
47. Melpignano, D., Benini, L., Flamand, E., Jego, B., Lepley, T., Haugou, G., Clermidy, F., Dutoit, D.: Platform 2012, a many-core computing accelerator for embedded SoCs. In: Proceedings of the 49th Annual Design Automation Conference on - DAC '12, p. 1137. ACM Press, New York, New York, USA (2012). DOI 10.1145/2228360.2228568. URL http://dl.acm.org/citation.cfm?id=2228568http://dl.acm.org/citation.cfm?doid=2228360.2228568
48. Merolla, P.a., Arthur, J.V., Alvarez-Icaza, R., Cassidy, a.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S.K., Appuswamy, R., Taba, B., Amir, A., Flickner, M.D., Risk, W.P., Manohar, R., Modha, D.S.: A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014). DOI 10.1126/science.1254642. URL http://www.sciencemag.org/cgi/doi/10.1126/science.1254642
49. Miro-Panades, I., Beigné, E., Thonnart, Y., Alacoque, L., Vivet, P., Lesecq, S., Puschini, D., Molnos, A., Thabet, F., Tain, B., Chehida, K.B., Engels, S., Wilson, R., Fuin, D.: A Fine-Grain Variation-Aware Dynamic Vdd-Hopping AVFS Architecture on a 32 nm GALS MPSoC. IEEE Journal of Solid-State Circuits **49**(7), 1475–1486 (2014). DOI 10.1109/JSSC.2014.2317137. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6807524
50. Moloney, D.: 1TOPS/W software programmable media processor. In: HotChips HC23. Stanford (2011)
51. Oh, J., Lee, S., Yoo, H.J.: 1.2-mW Online Learning Mixed-Mode Intelligent Inference Engine for Low-Power Real-Time Object Recognition Processor. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **21**(5), 921–933 (2013). DOI 10.1109/TVLSI.2012.2198249. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6210400
52. Park, S., Maashri, A.A., Irick, K.M., Chandrashekhar, A., Cotter, M., Chandramoorthy, N., Debole, M., Narayanan, V.: System-On-Chip for Biologically Inspired Vision Applications. IPSJ Transactions on System LSI Design Methodology **5**, 71–95 (2012). DOI 10.2197/ipsjtsldm.5.71. URL http://www.cse.psu.edu/~nic5090/HMAP/sldm.pdfhttp://japanlinkcenter.org/DN/JST.JSTAGE/ipsjtsldm/5.71?lang=en&from=CrossRef&type=abstract
53. Patterson, D.: The top 10 innovations in the new NVIDIA Fermi architecture, and the top 3 next challenges. NVIDIA Whitepaper (2009)
54. Pennebaker, W.B., Mitchell, J.L., Fogg, C., LeGall, D.: MPEG Digital Video Compression Standard. Chapman & Hall (1997)
55. Rahimi, A., Loi, I., Kakoee, M.R., Benini, L.: A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters. 2011 Design, Automation & Test in Europe pp. 1–6 (2011). DOI 10.1109/DATE.2011.5763085. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5763085
56. Rossi, D., Loi, I., Haugou, G., Benini, L.: Ultra-low-latency lightweight DMA for tightly coupled multi-core clusters. In: Proceedings of the 11th ACM Conference on Computing Frontiers - CF '14, pp. 1–10. ACM Press, New York, New York, USA (2014). DOI 10.1145/2597917.2597922. URL http://dl.acm.org/citation.cfm?doid=2597917.2597922
57. Rossi, D., Mucci, C., Campi, F., Spolzino, S., Vanzolini, L., Sahlbach, H., Whitty, S., Ernst, R., Putzke-Roming, W., Guerrieri, R.: Application Space Exploration of a Heterogeneous Run-Time Configurable Digital Signal Processor. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **21**(2), 193–205 (2013). DOI 10.1109/TVLSI.2012.2185963
58. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 pp. 1508–1515 Vol. 2 (2005). DOI 10.1109/ICCV.2005.104. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1544896
59. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. Computer VisionECCV 2006 pp. 1–14 (2006). DOI 10.1007/11744023_34. URL http://link.springer.com/chapter/10.1007/11744023_34
60. Sabarad, J., Kestur, S., Dantara, D., Narayanan, V., Khosla, D.: A reconfigurable accelerator for neuromorphic object recognition. In: 17th Asia and South Pacific Design Automation Conference, pp. 813–818. IEEE (2012). DOI 10.1109/ASPDAC.2012.6165067. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6165067
61. Scaramuzza, D., Achtelik, M., Doitsidis, L., Friedrich, F., Kosmatopoulos, E., Martinelli, A., Achtelik, M., Chli, M., Chatzichristofis, S., Kneip, L., Gurdan, D., Heng, L., Lee, G.H., Lynen, S., Pollefeys, M., Renzaglia, A., Siegwart, R., Stumpf, J., Tanskanen, P., Troiani, C., Weiss, S., Meier, L.: Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments. IEEE Robotics Automation Magazine **21**(3), 26–40 (2014). DOI 10.1109/MRA.2014.2322295
62. Seo, S., Dreslinski, R.G., Woh, M., Chakrabarti, C., Mahlke, S., Mudge, T.: Diet SODA: A Power-Efficient Processor for Digital Cameras. In: Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design - ISLPED '10, p. 79. ACM Press, New York, New York, USA (2010). DOI 10.1145/1840845.1840862. URL http://portal.acm.org/citation.cfm?doid=1840845.1840862
63. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: OverFeat: Integrated recognition, localization and detection using convolutional networks. arXiv:1312.6229 [cs] (2013). URL http://arxiv.org/abs/1312.6229. arXiv: 1312.6229
64. Toshev, A., Szegedy, C.: DeepPose : Human Pose Estimation via Deep Neural Networks. In: Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition (2014). DOI 10.1109/CVPR.2014.214
65. Wood, R.J., Finio, B., Karpelson, M., Ma, K., Pérez-Arancibia, N.O., Sreetharan, P.S., Tanaka, H., Whitney, J.P.: Progress on 'pico' air vehicles. The International

Journal of Robotics Research **31**(11), 1292–1302 (2012). DOI 10.1177/0278364912455073

66. Yoon, J.S., Kim, J.H., Kim, H.E., Lee, W.Y., Kim, S.H., Chung, K., Park, J.S., Kim, L.S.: A Unified Graphics and Vision Processor With a 0.89 uW/fps Pose Estimation Engine for Augmented Reality. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **21**(2), 206–216 (2013). DOI 10.1109/TVLSI.2012.2186157. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6156811`
67. Zhang, N., Paluri, M., Ranzato, M.A., Darrell, T., Bourdev, L., Berkeley, U.C.: PANDA : Pose Aligned Networks for Deep Attribute Modeling. In: Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition, vol. 2 (2014). DOI 10.1109/CVPR.2014.212