# Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing

Antonio Pullini, Davide Rossi, Igor Loi, Giuseppe Tagliavini, and Luca Benini

***Abstract*— This paper presents Mr.Wolf, a parallel ultra-low power (PULP) system on chip (SoC) featuring a hierarchical architecture with a small (12 kgates) microcontroller (MCU) class RISC-V core augmented with an autonomous IO subsystem for efficient data transfer from a wide set of peripherals. The small core can offload compute-intensive kernels to an eight-core floating-point capable of processing engine available on demand. The proposed SoC, implemented in a 40-nm LP CMOS technology, features a 108-$\mu$W fully retentive memory (512 kB). The IO subsystem is capable of transferring up to 1.6 Gbit/s from external devices to the memory in less than 2.5 mW. The eight-core compute cluster achieves a peak performance of 850 million of 32-bit integer multiply and accumulate per second (MMAC/s) and 500 million of 32-bit floating-point multiply and accumulate per second (MFMAC/s) −1 GFlop/s—with an energy efficiency up to 15 MMAC/s/mW and 9 MFMAC/s/mW. These building blocks are supported by aggressive on-chip power conversion and management, enabling energy-proportional heterogeneous computing for always-on IoT end nodes improving performance by several orders of magnitude with respect to traditional single-core MCUs within a power envelope of 153 mW. We demonstrated the capabilities of the proposed SoC on a wide set of near-sensor processing kernels showing that Mr.Wolf can deliver performance up to 16.4 GOp/s with energy efficiency up to 274 MOp/s/mW on real-life applications, paving the way for always-on data analytics on high-bandwidth sensors at the edge of the Internet of Things.**

***Index Terms*— Digital signal processors, dynamic voltage scaling, memory architecture, multicore processing, parallel architectures.**

## I. Introduction

The majority of current ultra-low-power smart-sensing edge devices operating for years on small batteries can handle only low-bandwidth sensors, such as temperature or pressure. The main design driver for these systems is to consume the smallest possible amount of power at the cost of performance, which is acceptable for most applications linked to low-bandwidth sensors [1]. In this direction, several approaches have been proposed to reduce as much as possible the power consumption of deeply embedded computing systems mainly focusing on the design of sub-threshold processors [2], [3] operating at frequencies ranges from a few tens of kHz up to a few MHz.

Some of the approaches for power minimization exploit partial shut-down strategies to leverage the heavy duty-cycled nature of these applications, developing systems able to keep the deep-sleep power of processors as low as a few tens of nW [1]. Other approaches exploit deep circuit-level optimizations, such as transmission-gate standard cells [4] and dual-mode standard cells, optimized for energy efficiency in the normal mode (NM) and for power consumption in the leakage-suppression mode (LSM), delivering sub-nW power consumption at the operating frequency of few Hz [5], [6].

While the low bandwidth generated by the aforementioned sensors allows to transmit raw data to the cloud for external analysis, a new generation of edge applications is emerging, which is focused on probing the environment with data-rich sensors (such as vibration, audio, video, or biopotentials sensors) and performing data-intensive computations locally at the sensors. This approach, preventing raw data to be transmitted wirelessly, is beneficial in terms of energy, aggregate sensor bandwidth, and security [7]. A possible way to tackle this challenge is to bring significant portion of the data analytics close to the sensor, reducing the high-bandwidth raw sensor output to highly compressed and informative data, such as tags, classes, or even simple trigger events or alarms. However, this approach poses an extreme challenge of squeezing the computational requirements of advanced near-sensor data analysis algorithms within the mW-range power envelope of always-on battery-powered IoT end nodes.

The solutions proposed during the last few years to deal with the increasing performance requirements of near-sensor data analytics applications mainly leverage two approaches. The first one lies in widening the operating range of low-power processors to target a higher peak performance while maintaining reasonable efficiency for low-performance applications [8]–[10]. However, when the performance constraints are so tight that the system is forced to operate out of the near-threshold region [11], energy efficiency unavoidably drops. The second approach lies in the system specialization. Extending end-node devices with accelerators dedicated to specialized functions can significantly improve energy efficiency for these specific tasks, while leveraging general purpose processors for other tasks. This approach

Manuscript received November 19, 2018; revised February 7, 2019; accepted April 9, 2019. This paper was approved by Guest Editor Stefan Rusu. This work was supported in part by the Swiss National Science Foundation (MicroLearn: Micropower Deep Learning) under Grant 162524 and in part by the Open transPREcision COMPuting (OPRECOMP) Project funded by the European Union's Horizon 2020 Research and Innovation Program under Grant 732631. *(Corresponding author: Antonio Pullini.)*

A. Pullini is with the Integrated Systems Laboratory, ETH Zürich, 8092 Zurich, Switzerland (e-mail: pullinia@iis.ee.ethz.ch).

D. Rossi and L. Benini are with the Integrated Systems Laboratory, ETH Zürich, 8092 Zurich, Switzerland, and also with the Department of Electrical, Electronic, and Information Engineering, University of Bologna, 40136 Bologna, Italy.

I. Loi and G. Tagliavini are with the Department of Electrical, Electronic, and Information Engineering, University of Bologna, 40136 Bologna, Italy.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2019.2912307

has been effectively adopted in secure artificial intelligence processors featuring convolutional neural network (CNN) accelerators for data analytics and crypto accelerators for security [12]–[14]. A much more flexible solution lies in parallel near-threshold computing. This approach exploits the energy benefits of near-threshold operations without compromising: 1) performance thanks to parallel execution over multiple cores and 2) flexibility leveraging software programmable processors [15].

In this scenario, one of the big challenges is to join low-power capabilities and energy efficiency of MCUs with peak performance of more complex architecture, such as digital signal processors (DSPs) and parallel processors. Indeed, while always-on IoT applications rely on ultra-low-power MCUs featuring modest compute capability cores, such as ARM Cortex M0 +, most of this new generation of applications require much more computational power [up to a few giga operations per second (GOp/s)] and significant memory footprint (up to a few MB). These requirements have to be achieved without compromising the tens-of-mW power envelope, coupled with state retentive deep sleep modes to deal with the heavily duty-cycled behavior of several IoT applications. Moreover, floating-point capable processors are desirable to ease the porting and to deal with the high dynamic range of some near-sensor data analytics applications, especially in the field of bio-potential processing.

To address this challenge, we propose Mr.Wolf, a multi-GOp/s fully programmable power/performance/precision-tunable IoT-edge computing engine fabricated in 40-nm LP CMOS technology. Mr.Wolf exploits the flexible attributes of the RISC-V ISA to deliver a state-of-the-art MCU called fabric controller (FC) coupled with a powerful programmable parallel processing engine for flexible multi-sensor (image, audio, bio-potentials, and inertial) data analysis and fusion. The system on chip (SoC) is built around an ultra-low power MCU subsystem based on a two-pipeline stage processor optimized for low power featuring a programmable 72–108-$\mu$W state-retentive sleep power (for up to 512 kB of system memory) and an I/O subsystem optimized for efficient and autonomous (with minimal processor intervention) data transfers from high-bandwidth peripherals (up to 1.6 Gbit/s aggregated bandwidth in 2.5 mW). The compute cluster is composed of eight fully programmable processors featuring DSP extensions targeting energy-efficient digital signal processing delivering up to 800 MMAC/s and up to 15 MMAC/s/mW, and sharing a floating-point unit (FPU) delivering up to 500 MFMAC/s (i.e., 1 GFlop/s) and up to 9 MFMAC/s/mW, when executing 32-bit fixed-point and 32-bit floating point matrix multiplication, respectively. We demonstrated the performance and efficiency of Mr.Wolf on a wide range of real-life applications belonging to audio, image, and bio-potential processing, showing that it can deliver performance from 1.1 to 16.4 GOp/s with energy efficiency from 18 to 274 MOp/s/mW.

The rest of this paper, extending the short abstract presented at ESSCIRC 2018 [16], is organized as follows. Section II introduces the Mr.Wolf SoC architecture, focusing on its key innovation aspects: autonomous IO, high-bandwidth L2 memory architecture, parallel computing accelerator, and power management. Section III describes the implementation of the SoC in 40-nm CMOS technology and presents the power/performance figures measured on the silicon prototype. Section IV presents the benchmarking with a wide set of near-sensor processing kernels and Section V provides a detailed comparison with respect to the state of the art. Finally, Section VI provides the conclusion with some final remarks.

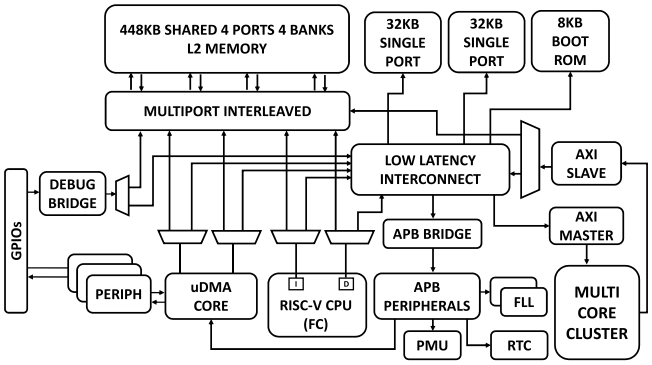


Fig. 1. Mr.Wolf SoC block diagram. Arrows indicate master-to-slave connections.

## II. Mr.Wolf SoC

Fig. 1 provides a top-level view of the Mr.Wolf architecture. It includes two power domains isolated by level shifters and dual-clock FIFOs to operate into independent voltage and frequency islands: the SoC and the cluster domains.

### A. SoC Subsystem

The SoC domain consists of an MCU built around a 12 kgates, two-pipeline stage RISC-V processor optimized for low power consumption (called zero-RISCY), referred to as FC, and 512 kB of L2 memory (Fig. 1). The processor implements the RV32IMC RISC-V ISA [17] and includes an integer 32-bit sequential multiplier featuring a latency of 3 cycles and an integer 32-bit divider featuring a latency of 35 cycles. This processor configuration was selected to optimize the tradeoff between power and performance in control-oriented tasks typical of a controller, such as IO management: it reduces the power consumption by a factor of 2 with respect to the DSP processors available on the cluster without compromising performance for tasks where scalar 32-bit arithmetic is needed (e.g., address manipulations, simple calculations, as used in IO drivers, and control code) [18].

The SoC has a full set of peripherals typical of advanced MCUs: Quad SPI (400 Mbit/s), I2C, 4 I2S ($4 \times 3$ Mbit/s), a parallel camera interface (400 Mbit/s), UART, four channel PWM interface, GPIOs, and a JTAG interface for debug purposes. The set of peripherals available on Mr.Wolf, together with the autonomous IO subsystem described in the following, enable parallel capture of images, sounds, and vibrations at high bandwidth and efficiency. The additional HyperBus peripheral available on Mr.Wolf allows to extend the on-chip memory by means of a DDR interface featuring 800 Mbit/s of bandwidth.

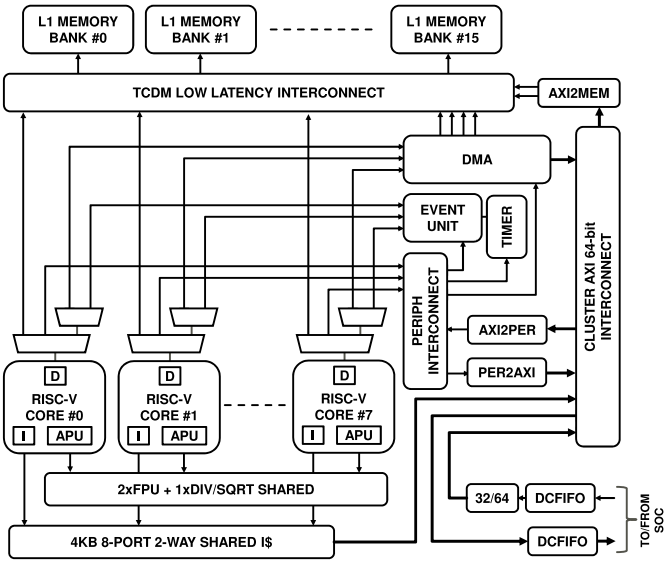


Fig. 2. Mr.Wolf parallel computing cluster. Arrows indicate master-to-slave connections.

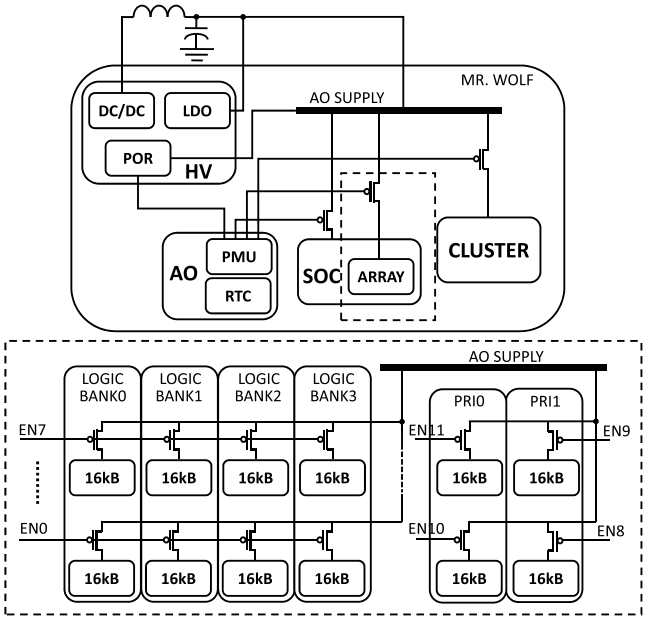


Fig. 3. Mr.Wolf power management and state-retentive L2 memory architecture.

Efficient sharing of on-chip memory resources is one of the key aspects to target high computing efficiency, since all functional units (CPU, IO peripherals, and parallel computing cluster) share data through the L2 memory, typically exploiting a double-buffering mechanism (i.e., data transfers from peripherals and L2 and from L2 to L1 memory are completely overlapped). To this end, in Mr.Wolf, the 512 kB of L2 memory are arranged as four 112-kB word-level interleaved logical banks (448 kB overall) on top of the two 32-kB private banks. This approach increases the access bandwidth to L2 memory by $4\times$, minimizing conflicts during parallel accesses through the six master ports of the L2 memory interconnect (i.e., the $\mu$DMA, the processor, and the cluster domain). Each logical bank is further split into eight physical memory banks (Fig. 3) that can be independently power gated, allowing to implement an incremental state-retentive mechanism for the L2 memory. The memory hierarchy of Mr.Wolf is organized as a single address space: every master in the chip can access all memory locations, easing the overall programmability of the system.

Private and low-latency accesses are needed for data and more importantly for instruction accesses coming from the FC. The FC does not have an instruction cache so when the CPU is active, the bandwidth on the instruction port is 3.2 Gb/s at 100 MHz. Such bandwidth, if directed to the shared memory, would significantly increase the contention ratio degrading the performance of both the FC and the other resources sharing their data through the L2. For this reason, two banks of 32 kB can be used privately by the FC (e.g., program, stack, and private data) without incurring any banking conflicts, and improving the performance of the FC by up to $2\times$ during execution of highly memory-intensive applications.

The connection with the parallel processing cluster consists of two asymmetric AXI plugs featuring a 64-bit width for cluster-to-memory communication and 32-bit for FC to cluster communication. The bus has been designed in an asymmetric way to save area, since the only master in the SoC domain (i.e., the FC) is only able to generate up to 32-bit blocking transactions (i.e., the FC is not able to generate bursts). Indeed, high-bandwidth data transfers are handled entirely by the DMA of the cluster through the 64-bit plug connected to the cluster AXI bus. Despite the high-performance interconnect, the SoC features a low-cost APB subsystem to access configuration registers of the different SoC IO peripheral IPs including pad GPIO and multiplexing control, clock and power control, timer, *$\mu$DMA* configuration port, and PWM controller.

Energy-efficient IoT systems require not only efficient processing engines but also an efficient I/O subsystem. Mr.Wolf implements an advanced I/O subsystem in which each peripheral has a dedicated lightweight DMA channel ($\mu$DMA) that enables the peripherals themselves to control the data transfer to/from the L2 memory. The $\mu$DMA has two dedicated 32-bit ports on the L2 memory interconnect, granting an aggregated bandwidth equal to $2 \times$ 32-bit $\times$ SoC clock frequency, sufficient to satisfy the requirements of parallel transfers from all the peripherals (up to 1.6 Gbit/s) with a frequency of just 57 MHz and a power of 2 mW. This architecture, coupled with the single-cycle latency multi-ported memory structure described above, guarantees to have a predictable latency to the memory. Moreover, it allows multiple concurrent data transfers toward external devices while operating at low frequency with no need for large buffers attached to the peripherals (16 B/channel are employed in Mr.Wolf). Some of the peripherals are equipped with an internal transaction engine that allows them to implement complex I/O transfers with completely autonomous synchronization between the involved I/O resources, so that the FC only duty is to set up the transaction and trigger its start.

### *B. Parallel Computing Cluster*

The cluster, shown in Fig.2, resides on a dedicated voltage and frequency domain, and is turned on and adjusted to the

required voltage and frequency when applications running on the FC offload highly intensive computation tasks.

It contains eight RISC-V cores supporting the RVC32IMF instruction set [17], plus an extension targeting energy-efficient digital signal processing (*Xpulp*) [19]. Such ISA extension consists of the first set of instructions, called *XpulpV1*, that can be easily inferred by compiler. This set of instructions include hardware loops to accelerate for statements typical of DSP kernels, load/store with post increment to accelerate incremental accesses to vectors and tensors, multiply and accumulate (MAC). A second set of extensions, called *Xpulpv2*, can be typically exploited inferring built-in intrinsic functions in the code, and include single instruction multiple data (SIMD) vectorial instructions, such as parallel arithmetic operations on 16-bit and 8-bit data, bit manipulation instructions useful to accelerate computations of emerging applications, such as binary neural networks (BNNs) [20], and support for fixed-point arithmetic, such as saturation and clipping. These extensions improve performance and energy efficiency of compute intensive kernels by up to 11×, if compared to a baseline RVC32IMF ISA (Fig. 10).

The cluster is served by a 64-kB multi-banked L1 scratchpad memory called tightly coupled data memory (TCDM) composed of 16 4-kB SRAM banks, enabling shared-memory parallel programming models such as OpenMP [21]. The L1 memory can serve all memory requests in parallel with single-cycle access latency, thanks to a low-latency logarithmic interconnect featuring a word-level interleaving scheme with round-robin arbitration resulting into a low average contention rate (<10% even on data-intensive kernels). A dedicated peripheral interconnect is used to access the cluster peripherals such as a timer and the event unit, as well as the AXI-4 bus. Data movements from L1 memory to L2 memory are explicitly managed by software through a lightweight DMA controller supporting 2-D addressing capabilities and up to 16 outstanding transactions toward the AXI-4 bus to hide access latency of the L2 memory. This approach has significantly smaller overhead with respect to an L1 data cache and allows to completely overlap data transfers and computation phases by means of double buffering.

The cluster program cache is implemented using latch-based memory to improve the access energy (by up to 4× for instruction memory [22]) of this high-bandwidth memory (25 Gbit/s at 100 MHz when all cores are active) with respect to traditional SRAM-based implementation. However, using latches instead of SRAMs comes at the cost of significant area overhead [22]. To reduce this overhead, taking advantage of the data-parallel computational model typically employed in the cluster, the instruction cache is shared among the cores, avoiding instruction replication on the private caches typically employed in traditional multi-core systems [23]. To ease the physical implementation of the latch-based memories and reducing routing congestion, the 4-kB instruction cache is split into four arrays. Each array has a single write port connected to the AXI-bus, used for refills, and eight read ports connected to the prefetch buffers of the RI5CY cores. This multi-port architecture allows the cores to have a non-blocking access to the cache with the same performances as a private cache. Moreover, refills are handled by a global controller so that if all cores are generating a miss accessing the same location, only one request is propagated to the L2 memory, reducing the pressure on the L2 memory. This approach couples the bandwidth (performance) benefits of private caches with the energy benefits of the latch-based implementation, mitigating the area overhead by means of sharing, improving energy efficiency by up to 1.5× with respect to an area-equivalent cluster architecture featuring a traditional SRAM-based private instruction cache.

Reducing parallelization overhead is one of the key elements for improving energy efficiency of computing systems relying on data-parallel computational models such as OpenMP [21], especially when dealing with applications characterized by unbalanced workloads and small parallel regions. Traditional parallel computing systems rely on software synchronization mechanisms (such as test-and-set) implemented through atomic instructions in dedicated memory regions. In Mr.Wolf, on top of the traditional software support, fast event management, parallel thread dispatching, and synchronization are supported by a dedicated hardware block (event unit) enabling low-overhead fine-grained parallelism to boost performance and energy efficiency of fine-grained parallel workloads. The processors can wait on generic events just performing a load operation on a register of the event unit mapped on a single-cycle aliased region accessed through the demuxes.

The event unit block also controls the top-level clock gating of every single core in the cluster, and hence a core waiting for an event (attached to a synchronization barrier or general event) is instantly brought into a fully clock gated state, zeroing its dynamic power consumption, and resumes the execution after the event in two clock cycles. When all the processors reach the synchronization of the barrier or the event is triggered, the event unit releases the clock gating of the processors that can resume the program flow. With respect to a traditional synchronization mechanism implemented with test-and-set instructions, the event unit reduces the latency and energy cost by up to 15× for barriers and by up to 1.5× for mutex, leading to a cluster-level speed-up (and energy) improvements up to 2×, during execution of applications with unbalanced workloads and small parallel regions such as Dijkstra, DWT, and FFT with respect to the execution of the same kernels with software barriers. To enable fast, non-blocking accesses, both the event unit and the DMA have dedicated ports to each CPU. The connection is made through a 2-level demuxing logic implemented close to the data port of the core. This design choice guarantees to prioritize access to timing critical low-latency interconnect over peripherals.

Floating-point capable processors are desirable in many deeply embedded applications [24], especially those dealing with processing of bio-potentials, often leveraging linear algebra algorithms featuring extremely high dynamic range [25], but also in other fields, such as audio and robotics [26]. Even when floating-point applications can be transformed into fixed point, this is not necessarily the best solution energy-wise, since the additional instructions required to deal with dynamic range of variables might incur significant overhead [27]. However, since FPUs are expensive in terms of area (the area of a

TABLE I
SUMMARY OF SHARED FPU FEATURES

| Unit | Latency | Pipelined/Iterative | # Of Shared Units |
|---|---|---|---|
| ADD | 2 | Pipelined | 1 |
| SUB | 2 | Pipelined | 1 |
| MUL | 2 | Pipelined | 1 |
| MAC | 3 | Pipelined | 2 |
| DIV | 4 | Iterative | 1 |
| SQRT | 6 | Iterative | 1 |
| CAST | 2 | Pipelined | 1 |

TABLE II
MR. WOLF POWER MODES

| Power Mode | VDD [V] | Frequency | Power |
|---|---|---|---|
| Deep Sleep* | 0.8 | n.a. | 72 $\mu$W |
| Ret. Deep Sleep* | 0.8 | n.a. | 76.5 – 108 $\mu$W |
| SoC Idle | 0.8 – 1.1 | SoC clock gated | 0.55 – 1.96 mW |
| SoC Active | 0.8 – 1.1 | 32 kHz – 450 MHz | 0.97 – 38 mW |
| Cluster Idle† | 0.8 – 1.1 | Cluster clock gated | 1.2 – 4.6 mW |
| Cluster Active‡ | 0.8 – 1.1 | 32 kHz – 350 MHz | 1.6 – 153 mW |

*From VBAT; †SoC must be active or idle; ‡SoC must be active;

full FPU is almost the same as the RI5CY core), their overhead needs to be minimized at the system level to deal with tight cost requirements of IoT end nodes. To address this challenge, the cluster implements a sharing approach for FPUs motivated by the following observations: 1) FPUs need to be pipelined to match the frequency of the rest of the system, featuring a latency of at least two cycles (Table I) and 2) the density of floating-point operations in applications is rarely larger than 50% because of load/store instructions needed to access the data, as highlighted in Table VI.

The FPU and two floating-point multiply and accumulate (FMACs) units are shared among the eight processors of the cluster. The FPU implements common floating-point operations summarized in Table I. The FPU and FMACs are integrated into the cluster through an auxiliary processing unit (APU) interconnect, featuring a request/grant protocol with round-robin arbitration as in the TCDM interconnect and communicating with the execute pipeline stage of the processor. Following this approach, each processor can transparently access each unit of the shared FPU, being stalled whenever the shared resource is used by another processor.

### C. Power Management

To maximize energy efficiency while minimizing the number of external components, the SoC contains a dual-mode voltage regulator composed of an internal dc/dc converter for active modes associated with a micro low-dropout (uLDO) regulator for low power modes, as shown in Fig. 3. The internal dc/dc converter can be directly connected to an external battery. It can deliver voltages in the range of 0.8–1.1 V when the circuit is active with an efficiency of 70% for very low loads (<500 $\mu$W) and up to 95% for medium and high loads (2–150 mW). When the circuit is in sleep mode, this regulator is turned off and the uLDO regulator is used to power the real-time clock fed by an off-chip 32-kHz crystal oscillator, which controls programmed wake-up and, optionally, part of the L2 memory allowing retention of application state for fast wake-up. In this way, the quiescent current of the voltage regulators is reduced to 290 nA. The wake-up is controlled by an embedded power manager that, depending on the system state, reboots the SoC from external memory (QSPI or HyperRAM) or from the L2 memory, supervising the power state transitions of the different domains.

Eight ultra-low leakage SRAM banks can be independently power gated, allowing to implement an incremental state-retentive mechanism for the L2 memory, where the leakage power of the memory arrays is incurred only for the memory banks that are actually being used by the applications. When in deep sleep, the current consumption is reduced to 72 $\mu$W (from VBAT) assuming the RTC is active and no data retention, and up to 108 $\mu$W assuming full L2 retention. The two main domains have their own separate clocks, generated by two frequency-locked loops (FLLs) placed on the SoC domain. Special attention has been paid to the time needed to turn on and turn off the cluster. The typical turn-around time from FC idle to cluster active is always below 300 $\mu$s allowing for agile power state transitions. Table II shows the power modes of Mr.Wolf together with maximum frequency and power consumption.

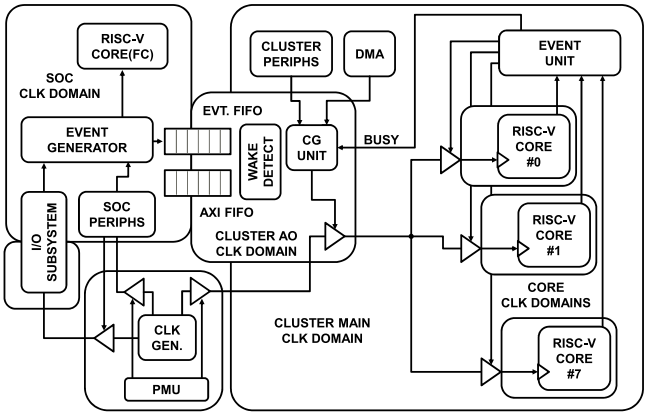


Fig. 4. Mr.Wolf clock domains and hierarchical clock gating.

Unlike other power modes, the Cluster Idle power mode is automatically activated in hardware. Fig. 4 shows an overview of the clock domains used in Mr.Wolf and how the hardware clock gating in the cluster works. Each IP in the cluster provides a busy signal to a central clock gating unit notifying the presence of pending transactions. When all resources are not busy and no transactions are in flight on the interconnect, the clock is gated at the source of the clock tree completely cutting the dynamic power of the cluster. The circuit is then reactivated in a single clock cycle when a transaction or an event arrives at the boundary of the cluster.

## III. MR. WOLF CHIP

### A. Implementation

Fig. 6 shows a die photograph and the floorplan of Mr.Wolf, while Fig. 5 provides a detail of the layout of the SoC. The SoC was implemented in TSMC 40-nm CMOS LP technology.

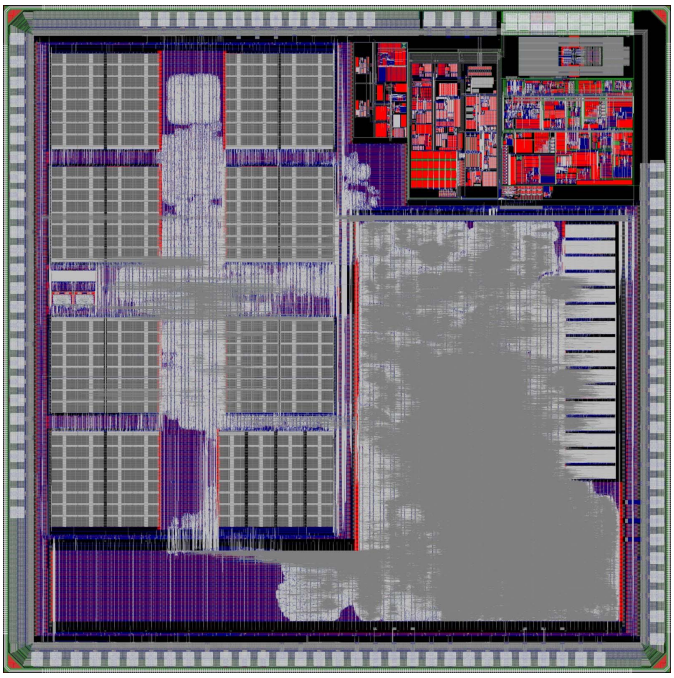

Fig. 5. Mr.Wolf SoC layout.

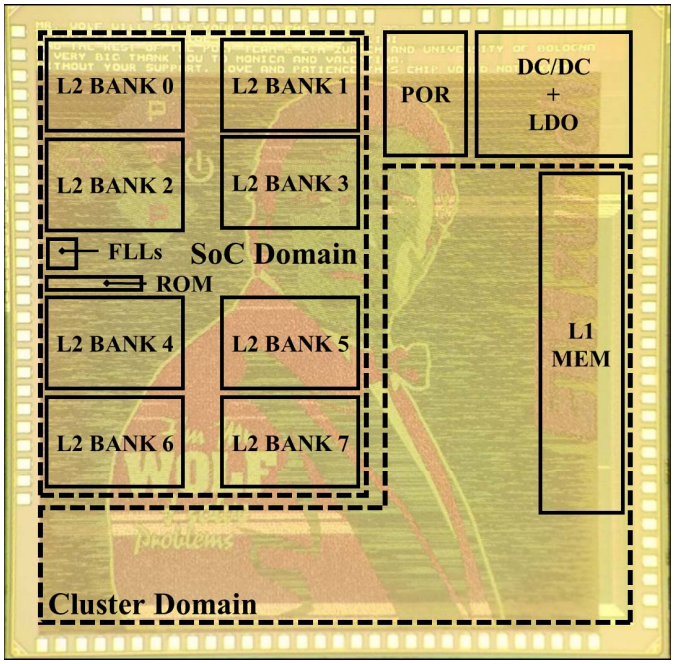


Fig. 6. Mr.Wolf. SoC micrograph and floor plan description.

TABLE III
MR. WOLF SOC FEATURES

| | |
|---|---|
| Technology | CMOS 40nm LP |
| Chip Area | 10mm$^2$ |
| Memory Transistors | 576 kB |
| Equivalent Gates (NAND2) | 1.8 Mgates |
| Voltage Range | 0.8 V – 1.1 V |
| Frequency Range | 32 kHz – 450 MHz |
| Power Range | 72 $\mu$W – 153mW |

It was synthesized with Synopsys Design Compiler 2016.12, while Place & Route was performed with Cadence Innovus 16.10. The two main power domains of the chip (SoC and cluster domains) are highlighted in Fig. 6 with dashed lines, while the power on reset (POR) and dc/dc converter have been placed in the third, always-on power domain. Both SoC and power domains are switchable. The power switches residing in the two domains are supplied by the dc/dc and controlled digitally by the power manager placed in the always-on domain to selectively turn on each domain. The output of the power switches is then connected to the rails powering the cells and memories of each domain.

In order to enable selective retention mode for each of the L2 memory banks highlighted in Fig. 6, a power ring supplied by the always-on VDD (0.8 V) was placed around each bank, and a few additional power switches were placed inside each sub-domain to selectively supply the array of the SRAM banks when the SoC domain is off. This selective state-retention mode has an area overhead of approximately 2% of the overall chip area. The two FLLs and the bootup ROM reside in the SoC domain as well.

Table III summarizes the main features of Mr.Wolf SoC. The die size is 10 mm$^2$, integrating 1.8 million of equivalent logic gates (minimum sized NAND2) and 576 kB of memory, and featuring a voltage range of 0.8–1.1 V and an operating frequency ranging from 32 kHz and 450 MHz, while the power consumption ranges from 72 $\mu$W to 153 mW.

Table IV highlights the main contributions to the overall chip area. The three largest components of the SoC are the L2 memory subsystem (i.e., 512 kB of multi-banked L2 memory + SoC interconnect), the dc/dc and POR, and the cluster, while the combined contribution of the IO and FC subsystems is smaller than 4%. Within the compute cluster (Table V), 27% of the area is used by the eight RI5CY processors, while 16.3% of the area is occupied by the 16-port 64-kB multi-banked TCDM, implemented by 16 $1024 \times 32$-bit SRAM banks. A relevant amount of the cluster area (38.9%) is used by the 4-kB shared instruction cache, implemented with a standard cell-based (i.e., latches) approach. Although the latch-based implementation features a significant area overhead with respect to more traditional approaches based on SRAMs, it provides major energy savings [22], which is one of the main reasons for the significantly higher energy efficiency of the cluster with respect to the FC, as discussed in Section III-B. The FPUs occupy only 4.3% of the area, thanks to the sharing approach adopted in the cluster architecture, saving significant area with respect to a private FPU approach for which 30% area overhead would be needed. Finally, the remaining area is used by smaller blocks, such as DMA and event unit, and by the interconnect.

### *B. Performance and Energy Efficiency*

Fig. 7 shows the SoC performance measured on the silicon prototype running a typical high-utilization workload (matrix multiplication), while Fig. 8 shows the related energy efficiency. The first two curves (blue and red) show the FC and cluster performance when executing an integer matrix

TABLE IV
MR.WOLF SOC AREA BREAKDOWN

| Instance | Area [$\mu$m2] | Percentage [%] |
|---|---|---|
| Pad Frame | 1516800 | 15,17 |
| DC/DC | 991900 | 9,92 |
| PorBor | 119600 | 1,20 |
| SoC Domain | 3240900 | 32,41 |
| Safe Domain | 1169259 | 11,69 |
| Cluster Domain | 2961541 | 29,62 |

TABLE V
MR.WOLF CLUSTER AREA BREAKDOWN

| Instance | Area [$\mu$m2] | Percentage [%] |
|---|---|---|
| Cores | 400000 | 26,9 |
| TCDM | 242508 | 16,3 |
| Interconnect | 41501 | 2,8 |
| Bus AXI | 37952 | 2,6 |
| I$ | 579239 | 38,9 |
| DMA | 81381 | 5,5 |
| Event Unit | 41318 | 2,8 |
| FPU | 63575 | 4,3 |

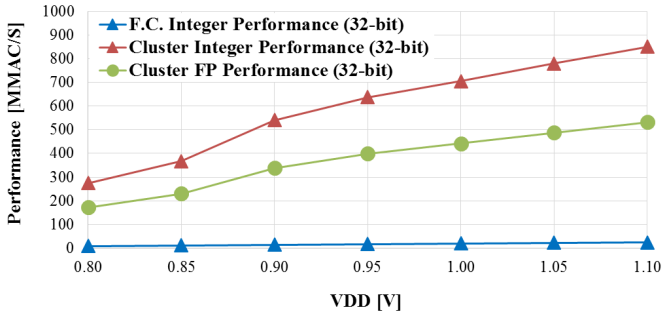


Fig. 7. Mr.Wolf performance when executing an integer matrix multiplication on the FC and on the cluster and a floating-point matrix multiplication on the cluster.

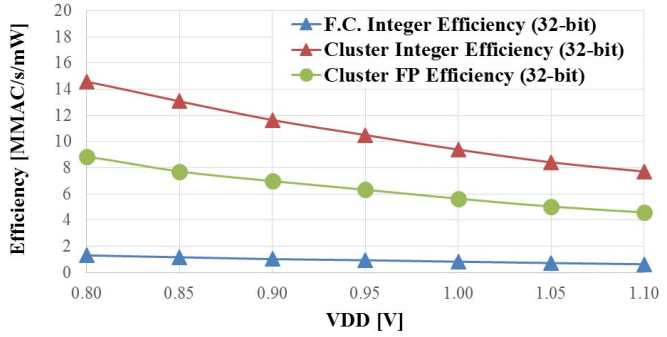


Fig. 8. Mr.Wolf energy efficiency when executing an integer matrix multiplication on the FC and on the cluster and a floating-point matrix multiplication on the cluster.

multiplication. It is possible to note that similar to other low-power MCUs [3], [28], based on tiny processors optimized for low-power control tasks, the FC can achieve a peak performance of 25 MMAC/s at 450 MHz, 1.1 V, and a peak efficiency of 1.5 MMAC/s/mW at 150 MHz, 0.8 V. The differentiating factor and the power of Mr.Wolf stand on the possibility to power-on the parallel processing cluster and offload compute-intensive tasks with significant performance and efficiency. Thanks to the instruction set extensions, the optimized pipeline of the eight RI5CY processors, and most importantly to the efficient memory sharing through the L1 data memory and instruction cache, the cluster can execute 2.5 MAC/cycle on eight cores. This execution efficiency leads to the peak performance of 850 MMAC/s at 350 MHz, 1.1 V and a peak energy efficiency of 15 MMAC/s/mW at 110 MHz, 0.8 V, improving DSP performance and energy efficiency with respect to the FC by 35× and 12×, respectively. The third curve (green) shows the performance and efficiency of the cluster when executing an FP matrix multiplication, expressed as MFMAC/s and MFMAC/s/mW, respectively. It is interesting to note that even if the FMAC units are shared, and despite their two pipeline stages (required to reach the target frequency), the architectural efficiency is 1.57 FMAC/cycle, leading to a peak performance of 500 MFMAC/s—1 GFlop/s—and a peak energy efficiency of 9 MFMAC/s/mW.

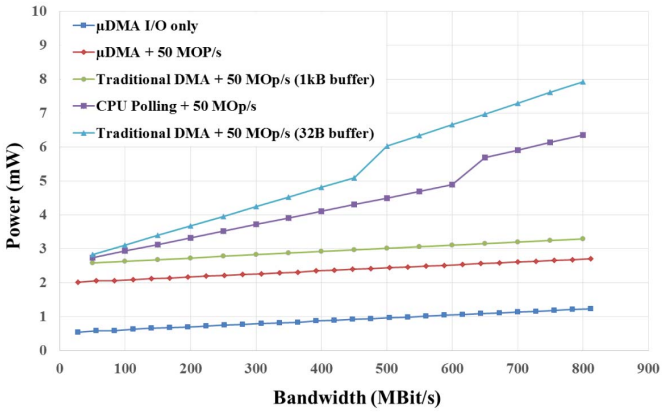


Fig. 9. Mr.Wolf IO subsystem efficiency.

### C. IO Performance

Fig. 9 shows in the lower two curves, the measurements of power consumption of the SoC subsystem for different I/O input bandwidths with and without 50 millions of operations per second (MOp/s) of load on the FC. The system can sustain 800 MBit/s operating at as low as 28.5 MHz and consuming 1.21 mW when only I/O to memory transfer is involved and work at 62.5 MHz and consume 2.7 mW when executing a kernel on the CPU at 50 MOp/s in parallel to the I/O transfer. The other curves in the graph show an estimate of the consumption of a more traditional system with CPU, memory, and DMA sharing the same system bus. When DMA is used, performance is deeply affected by the buffer size.

The two extremes are shown in the graph: a small buffer of 32 B and a hypothetical system with 1-kB buffer dedicated to each single peripheral. The proposed solution shows an improvement of up to 4.5× in power consumption when operating at high bandwidth (800 MBit/s) compared to a traditional DMA solution with a small 32 B/peripheral buffer. Moreover, even when very big buffers are employed in traditional architectures, the $\mu$DMA is still 1.2× more efficient for three reasons: 1) traditional architectures are required to use higher frequency to compensate for contention with the processor

on the system memory; 2) tightly coupled integration with the L2 memory enables the use of smaller (and more energy efficient) buffers; and 3) in traditional architectures, the whole system interconnect is active during I/O transfers (e.g., AHB matrix).

In Fig. 9, CPU polling is presented as a reference and to highlight that in a traditional DMA-based solution, the buffering resources at the peripherals have to be big enough to hide the overhead of the DMA interrupt service routine and DMA programming. Moreover, in a traditional system, buffering has to be allocated at design time and cannot be dynamically allocated as in our solution.

## IV. Mr.Wolf Benchmarking

This section presents an extensive architectural evaluation of the Mr.Wolf SoC, when executing parallel kernels belonging to different near-sensor processing application fields (audio processing, image processing, and bio-potentials). The set of kernels selected for benchmarking, presented in Table VI, forms the building blocks of several real-life applications, such as EMG-based gesture recognition [29], seizure detection [30], object detection [20], and many others, being the ones where most of the time (and energy) is typically spent in the considered applications. The selected set of kernels is highly heterogeneous (i.e., we did not choose variations of the same basic pattern) to emphasize the flexibility of the architecture. To evaluate the performance of Mr.Wolf, the kernels were executed in isolation to assess the capability of the system to deal with arithmetic operations with different operands bit-width and precision: floating-point, 32-bit, 16-bit, and 8-bit fixed point, and bit-wise. All the benchmarks were compiled with a version of GCC 7.1 enhanced to support *Xpulp* extensions (discussed in Section II) and parallelized with a runtime library optimized for PULP architectures.

All the benchmarks were executed on an evaluation board hosting a prototype of Mr.Wolf connected to a PC through a JTAG adapter for loading binaries to the L2 memory. Table VI summarizes the benchmark set and their relevant features, such as precision, code size, and floating-point density (i.e., the number of floating-point instructions versus the overall number of instructions executed by the processors). Moreover, Table VI reports the main results of their implementation and optimization in Mr.Wolf, namely: 1) the instruction per cycle (IPC) on the eight-core cluster (referred to the RV32IMFCXpulp instructions actually executed by the cores); 2) the % of stalls extracted from the performance counters available on the RI5CY processors; and 3) the performance and energy efficiency, normalized to equivalent RV32IMC operations. For a fair comparison with respect to a sequential implementation, the overhead of the parallel runtime (e.g., additional instructions for thread dispatching and synchronization) is taken into account during the parallel execution but not during the normalization with respect to the equivalent RV32IMC operations used to derive performance and efficiency.

Fig. 10 reports the benchmarking of fixed-point kernels highlighting the performance boost when moving the execution from the FC of Mr.Wolf (i.e., a zero-RISCY processor) to the cluster (single-core execution on an RISCY processor), and when exploiting the *Xpulp* extensions. A detailed description of the instruction set extensions of RI5CY core can be found in [19]. It is possible to note that the performance increases from 1.1× to 1.9× when moving from zero-RISCY to RI5CY due to the pipeline optimized for energy-efficient digital signal processing featuring single-cycle multiplication and load/store operations. A further performance boost can be noted when enabling the Xpulp extensions on the RI5CY processor. While some of the extensions are general purpose and are automatically inferred by the compiler enabling the related flags (*XpulpV1*), some of the benchmarks have been optimized through the usage of intrinsic functions (also known as built-in functions) in the source code of the applications to better exploit the capabilities of the underlying hardware (*Xpulpv2*). This effect can be noted in Fig. 10 when analyzing applications featuring smaller than 32-bit precision such as *MatMul*, *5x5 Conv*, and *CNN Layer*, exploiting the vectorial capabilities of the processor able to execute two 16-bit operations and four 8-bit operations in parallel. When exploiting compiler optimization, the speed-up with respect to zero-RISCY ranges from 2.3× to 3.9×, which is further boosted when exploiting manual optimization with intrinsic functions, leading to speed-ups ranging from 4.4× to 11.7×. Applications leveraging bit-manipulation instructions, such as bit-insert, bit-extract, and pop-count, also significantly benefit from the exploitation of intrinsic functions, featuring a speed-up up to 3.5× with respect to the execution on zero-RISCY leveraging shift and mask operation to handle bitwise computations.

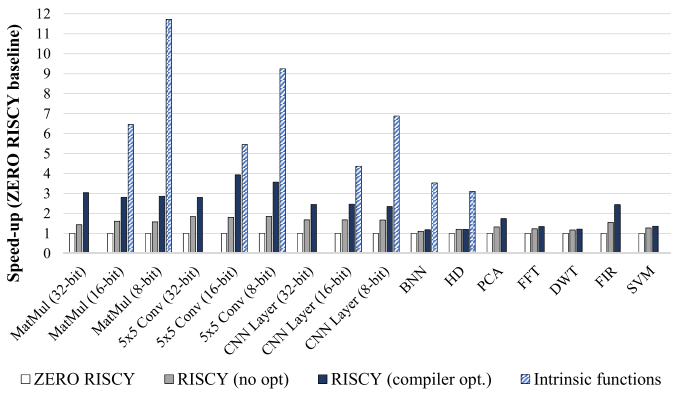


Fig. 10. Benchmarking of zero-RISCY and RI5CY processors and benefit of *XpulpV1* and *XpulpV2* extensions on a set of fixed-point near-sensor processing kernels.

Fig. 11 shows the performance boost achieved by the cluster when executing the fixed-point benchmarks on two, four, and eight cores, where benchmarks are sorted from the lowest to the highest speed-up. The overlying hollow bars depict Amdahl's limit of each application, which is the maximum speed-up theoretically achievable by the parallel execution. Table VI shows other relevant information related to the 8-parallel core implementation, such as the number of stalls due to contention in TCDM, the number of load-use stalls, the number of I$ stalls, and the number of IPCs delivered

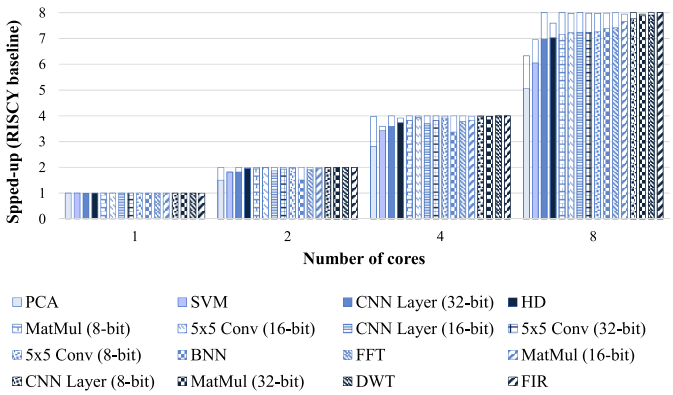


Fig. 11. Speed-up of benchmarks with respect to the number of active cluster cores. Hollow bars: Amdahl's limit of applications. Bold bars: speed-up on Mr.Wolf.

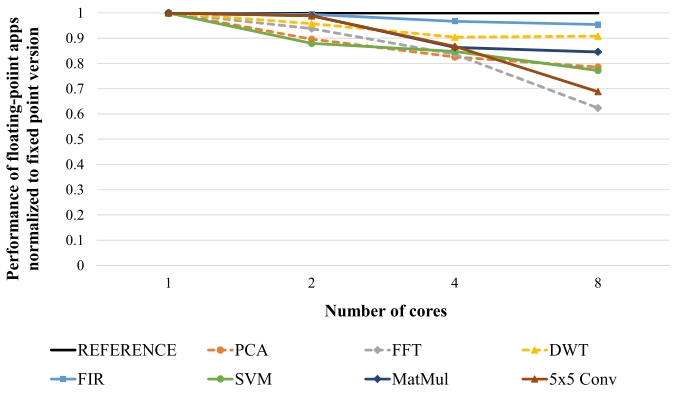


Fig. 12. Floating-point efficiency compared to the fixed-point version of the benchmarks.

by the cluster. It is possible to note that most highly parallel applications (i.e., featuring Amdahl's limit close to 2, 4, and 8) feature almost linear speed-up. In general, the performance drop with respect to the ideal speed-up is smaller than 15% even for applications featuring very small parallel regions and frequent synchronization barriers, such as FFT, and highly unbalanced workloads such as PCA. This is achieved mainly thanks to the efficient data sharing implemented by the TCDM interconnect, which keeps the worst case access contention to the shared memory banks to 7% even on an extremely LD/ST intensive workload such as 32-bit $5 \times 5$ convolution ($5 \times 5$ *Conv* 32-bit), the 4-kB shared instruction cache architecture which perfectly fits the program memory footprint of near-sensor processing applications, allowing to keep the stalls related to instruction fetch below 1% also for large applications (such as PCA and BNN, as reported by the code size in Table VI), and the efficient hardware-assisted synchronization mechanism discussed in Section II. The overall performance of the cluster, normalized to equivalent RV32IMC operations, ranges between 1.1 GOp/s more and 16.4 GOp/s; the higher values are achieved when the 8-bit SIMD extensions can be fully exploited on highly parallel workload, such as *MatMul*, $5 \times 5$ *Conv*, or CNN layer.

Fig. 12 shows the reduction of performance due to the sharing of FPUs when increasing the number of cores from 1 to 8. The floating-point efficiency is highly related to the floating-point operation density (i.e., the number of floating-point operations normalized to the total number of executed operations), as summarized in Table VI. Applications featuring floating-point densities below 20% feature a high floating-point efficiency (above 90%) even when running on eight cores. Applications with higher density of floating-point operations feature high efficiency despite a floating-point efficiency down to 0.8%, with the exceptions of FFT and $5 \times 5$ *Conv*, featuring extremely high floating-point density (33% and 36%, respectively).

These results confirm the benefits of the sharing approach, providing floating-point capabilities to the parallel processing cluster with significantly smaller overhead than a private FPU approach, and leading to a performance overhead smaller than 20% for most of the applications. Overall, the floating-point performance of Mr.Wolf ranges between 1.1 and 2.9 GOp/s, considering operations normalized to equivalent RV32IMFC instructions, as summarized in Table VI.

## V. Comparison With State of the Art

Table VII shows a comparison with the devices defining the boundaries of the Mr.Wolf design space: low-power microcontrollers (MCUs) [1], [3], [28], wide-performance range DSPs [31]–[33], and PULP architectures [12], [13], [15]. Performance of existing architectures is normalized to equivalent RV32IMC instructions. Since, for most architectures, IPC is not reported, we have considered one IPC (IPC $=$ 1) and applied a $2.5\times$ factor to FULMINE, GAP-8, and Mr.Wolf MOp/s to take into account the performance boost of *Xpulp* extensions on near-sensor processing applications (Table VI).

With respect to tiny MCUs such as SleepWalker and REISC, Mr.Wolf delivers, orders of magnitude, better peak performance, and also $1.5\times$ better energy efficiency, despite the implementation strategy of these MCUs is highly optimized to operate at very low voltage (i.e., down to 0.4 V). Indeed, similar to the one of the zero-RISCY cores, the micro-architecture of these processors is not optimized for energy-efficient digital signal processing. Moreover, the architectures of these SoCs are designed with extremely simplified memory hierarchy (i.e., instruction and data memory of a few kB) which consume significantly less power than Mr.Wolf, but pose severe limitations during the execution of complex near-sensor data analytic applications. The superior energy efficiency of the Mr.Wolf cluster is determined by the optimized micro-architecture of the core (including DSP extensions) and coupled with the architectural features of the cluster, namely: 1) the efficient data memory banks' sharing; 2) latch-based shared instruction cache; and 3) efficient hardware synchronization management. All these features allow to achieve almost linear speed-ups for parallel execution of several applications (Table VI), with negligible power overhead, boosting the energy efficiency of the Mr.Wolf cluster. For instance, [28] reports an IPC of 0.63 for FFT and FIR (single core), while the Mr.Wolf cluster delivers 0.88 and 1 per core, respectively. The speed-up of the Mr.Wolf cluster with respect to basic RISC ISAs such as [28] jumps to $1.5\times$ and $3\times$ per core when considering XpulpV1 extensions of RI5CY, up to $10\times$ per core when exploiting

TABLE VI
MAIN FEATURES AND RESULTS OF APPLICATIONS USED FOR BENCHMARKING MR.WOLF SOC

| *Application* | *Domain* | *Precision* | *Code size* [bytes] | *F.P. Density* [%] | *IPC* | *TCDM Stalls* [%] | *LD-use Stalls* [%] | *I$ Stalls* [%] | *Performance*[a,b] [GOp/s] | *Efficiency*[a,c] [MOp/s/mW] |
|---|---|---|---|---|---|---|---|---|---|---|
| PCA | ExG | 32-bit F.P. | 3180 | 18 | 5.4 | 0.4 | 0.0 | 0.5 | 1.7 | 20 |
| | | 32-bit Int. | 6832 | – | 5.6 | 0.0 | 0.4 | 0.1 | 1.8 | 30 |
| FFT | Audio, Image, ExG | 32-bit F.P. | 2832 | 33 | 4.7 | 0.9 | 0.1 | 0.5 | 1.6 | 26 |
| | | 32-bit Int. | 4438 | – | 7.1 | 2.6 | 0.2 | 0.6 | 2.2 | 36 |
| DWT | Audio, Image, ExG | 32-bit F.P. | 698 | 8 | 5.4 | 2.9 | 2.9 | 0.3 | 1.9 | 31 |
| | | 32-bit Int. | 1274 | – | 5.2 | 2.2 | 1.7 | 0.7 | 2.0 | 33 |
| FIR | Audio, ExG | 32-bit F.P. | 490 | 19 | 7.0 | 1.2 | 0.0 | 0.2 | 3.7 | 62 |
| | | 32-bit Int. | 502 | – | 8.0 | 0.0 | 0.0 | 0.2 | 3.8 | 64 |
| SVM | ExG | 32-bit F.P. | 1288 | 16 | 6.1 | 0.6 | 0.0 | 0.6 | 1.1 | 18 |
| | | 32-bit Int. | 1448 | – | 5.2 | 0.5 | 0.0 | 0.5 | 1.3 | 21 |
| MatMul | Audio, Image, ExG | 32-bit F.P. | 1618 | 25 | 6.1 | 2.1 | 0.0 | 0.1 | 2.9 | 49 |
| | | 32-bit Int. | 1630 | – | 7.2 | 1.0 | 7.0 | 0.2 | 4.6 | 76 |
| | | 16-bit Int. | 704 | – | 7.5 | 2.1 | 1.8 | 0.4 | 9.9 | 164 |
| | | 8-bit Int. | 596 | – | 6.8 | 6.9 | 2.9 | 0.8 | 16.4 | 274 |
| 5x5 Conv | Audio, Image, ExG | 32-bit F.P. | 746 | 36 | 4.7 | 0.9 | 0.0 | 0.1 | 1.7 | 28 |
| | | 32-bit Int. | 588 | – | 7.0 | 7.0 | 5.5 | 0.2 | 3.7 | 62 |
| | | 16-bit Int. | 764 | – | 6.3 | 1.2 | 0.0 | 0.3 | 7.2 | 120 |
| | | 8-bit Int. | 500 | – | 5.6 | 0.6 | 0.1 | 0.6 | 12.2 | 203 |
| CNN Layer | Image | 32-bit Int. | 778 | – | 7.5 | 3.4 | 0.0 | 0.1 | 3.2 | 53 |
| | | 16-bit Int. | 1072 | – | 5.6 | 1.5 | 0.0 | 0.3 | 6.0 | 99 |
| | | 8-bit Int. | 842 | – | 5.0 | 1.0 | 0.0 | 0.4 | 10.0 | 167 |
| HD [29] | ExG | binary | 3326 | – | 7.1 | 0.6 | 0.8 | 0.2 | 7.7 | 129 |
| BNN [20] | Audio, Image | binary | 4204 | – | 7.1 | 0.3 | 2.8 | 2.4 | 5.8 | 97 |

[a] Equivalent RV32IMC and RV32IMFC operation are reported for fixed-point and floating-point applications, respectively.
[b] Performance at 350 MHz, 1.1V is reported.
[c] Efficiency at 110 MHz, 0.8V is reported.
[d] Includes code size of the kernels. Calls to external libraries (e.g. math standard functions) are not included.

TABLE VII
COMPARISON BETWEEN *Mr.Wolf* AND OTHER EMBEDDED SOCS REPRESENTATIVE OF THE STATE-OF-THE-ART IN LOW-POWER MCUS, WIDE PERFORMANCE RANGE DSPS, AND PULP ARCHITECTURES

| | | *Tech.* | *ISA* | *# of Cores* | *I$/D$/L2* [kB] | *Sleep Power* [$\mu$W] | *VDD Range* [V] | *Max Freq.* [MHz] | *Peak Int. Perf.*[a] [MOp/s] | *Peak Int. Eff.* [MOp/s/mW] | *Peak F.P. Perf.* [MFlop/s] | *Peak F.P. Eff.* [MFlop/s/mW] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *MCUs* | SleepWalker [3] | 65nm | 16-bit MSP430 | 1 | 16/2/n.a. | 1.5 st.ret. | 0.4 | 25 | 25 | 81.5[d] | - | - |
| | Myers et.al. [1] | 65nm | 32-bit Cortex-M0+ | 1 | n.a./n.a./24 | 0.08 st.ret. | 0.2–1.2 | 66 | 66 | 85 | - | - |
| | REISC [28] | 55nm | 32-bit | 1 | 8/8/n.a. | n.a. | 0.54–1.2 | 82.5 | 82.5 | 98 | - | - |
| *DSPs* | Hexagon [31] | 28nm | 32-bit VLIW | 1 | 16/32/256 | n.a. | 0.6–1.05 | 1200 | 3690[b] | 53[b] | - | - |
| | FRISBEE [32] | 28nm | 32-bit | 1 | 4/4/n.a. | n.a. | 0.4–1.3 | 2600 | 2600 | 16.1 | - | - |
| | RISC-V VP [33] | 28nm | RV64IMFC | 1 | n.a./n.a./n.a. | n.a. | 0.45–1 | 961 | 961 | 68[d] | 1900[e] | 68[d] |
| *PULP SoCs* | PULPv2 [15] | 28nm | OR32 | 4 | 4/48/64 | 200 | 0.32–1.15 | 825 | 3300 | 193 | - | - |
| | FULMINE [13] | 65nm | OR32 Xpulp | 4 | 4/64/192 | 120 | 0.8–1.1 | 400 | 4200[c] | 69 | - | - |
| | GAP-8 [12] | 55nm | RV32IMC Xpulp | 1+8 | 4/64/512 | 3.6–30 st.ret. | 1–1.2 | 250 | 3500[c] | 50 | - | - |
| | **Mr.Wolf (This Work)** | **40nm** | **RV32IMFC Xpulp** | **1+8** | **4/64/512** | **72–108 st.ret.** | **0.8–1.1** | **450** | **7000**[c] | **120** | **1000**[e] | **18** |

[a] Equivalent RVC32IM operations.
[b] An Efficiency of 80% is considered as upper bound for a 4-lanes VLIW, equivalent to an IPC of 3.2 [34].
[c] Considers performance ratio between execution with and without Xpulp extensions in ideal conditions (i.e. no stalls). 16-bit and 8-bit SIMD operations are not considered.
[d] Power density is normalized to 32-bit operations.
[e] Considering 1 MAC = 2 ops where MOp/s are reported, when executing a matrix multiplication.

the full XpulpV2 extensions, which include SIMD 16-bit, 8-bit instructions and bit-manipulation extensions (Table VI). Finally, for a given performance target (MOp/s), exploiting parallelism allows to achieve the same performance at a lower supply voltage, improving energy efficiency with respect to sequential execution. Similar considerations can be done for [1] and [3], leveraging ARM Cortex M0 + and 16-bit MSP430 ISAs, respectively.

Mr.Wolf also surpasses the performance of all existing wide-range DSPs (by more than 2×) with significant energy efficiency margin (more than 1.8×), when considering 32-bit operations (Table VII). Both performance and efficiency can be further increased on Mr.Wolf when exploiting SIMD instructions available on the Xpulp extensions and not available in other cores, leading to a performance and efficiency boost of 1.9× to 2.1× and 3.2× to 3.5×, when operating on 16-bit and 8-bit data, respectively (Table VI). The RISC-V vector processor [33] performs with 3.7× better energy efficiency than Mr.Wolf on floating-point workloads (normalized to 32-bit floating-point operations for fair comparison) [33],

thanks to the more scaled technology node (28-nm FD-SOI) that allows to operate at high frequency down to 0.45 V, and the architecture highly specialized for floating-point computations. However, the fixed-point performance and efficiency of the scalar RISC-V processor are significantly smaller, especially when enabling the SIMD extensions for smaller than 32-bit operations on Mr.Wolf. Finally, none of the described mobile processors feature state-retentive deep-sleep modes to enable duty-cycled operations for IoT applications.

Exploiting the heterogeneous architecture which couples the IO efficiency and state-retentive deep-sleep capabilities of the SoC domain with the powerful and energy-efficient 8-processor cluster, Mr.Wolf represents a significant advance in the state of the art of PULP processors. The efficiency of Mr.Wolf is surpassed only by PULPv2 (even though PULPv2 does not support the *Xpulp* ISA extensions) due to a more scaled technology used for implementation (28-nm FD-SOI versus CMOS 40-nm LP). However, PULPv2 is lacking internal power management circuits (i.e., dc/dc, LDO, and power gating), significantly decreasing the system-level efficiency for duty-cycled applications. Although low-power processors, such as Sleepwalker, [3] feature a better deep sleep power, GAP-8 and Mr.Wolf have the capability to store in a full retentive way up to 512 kB of data (instead of a few kB). However, while GAP-8 is more specialized for CNN workloads, featuring a dedicated accelerator, Mr.Wolf is 2.4× more efficient on fixed-point workloads and more general purpose thanks to the presence of the shared FPUs.

Finally, thanks to its autonomous IO subsystem and the hierarchical and energy-proportional architecture, Mr.Wolf allows to periodically wake-up the SoC only to efficiently transfer sensor data to L2 with the $\mu$DMA, accumulate data on the state-retentive L2 memory (enabling retention only on used banks to minimize sleep power), and activate the cluster when enough data has been acquired for energy-efficient (floating-point) digital signal processing, paving the way for always-on data analytics of high-bandwidth sensor data at the edge of the Internet of Things.

## VI. Conclusion

We presented Mr.Wolf, an SoC for edge IoT applications coupling a state-of-the-art MCU featuring an advanced IO subsystem for efficient data acquisition from high-bandwidth sensors, with an 8-core floating-point capable computing cluster. The proposed SoC, implemented in a commercial 40-nm technology, features a 108-$\mu$W fully retentive memory (512 kB), an efficient IO subsystem capable to transfer up to 1.6 Gbit/s in less than 2.5 mW, and an 8-core compute cluster achieving a peak performance of 850 MMAC/s and 500 MFMAC/s (1 GFlop/s) and an energy efficiency up to 15 MMAC/s/mW (and 9 MFMAC/s/mW). We demonstrated that Mr.Wolf SoC allows to perform parallel floating-point digital signal processing within a power envelope smaller than high-performance MCU. We demonstrated the capabilities of the proposed SoC on real-life near-sensor processing applications, showing that Mr.Wolf can deliver performance up to 16.4 GOp/s with energy efficiency up to 274 MOp/s/mW.

## Acknowledgment

The authors would like to thank Dolphin Integration for providing the Retention Alternating Regulator (RAR).

## References

[1] J. Myers, A. Savanth, R. Gaddh, D. Howard, P. Prabhat, and D. Flynn, "A subthreshold ARM cortex-M0+ subsystem in 65 nm CMOS for WSN applications with 14 power domains, 10T SRAM, and integrated voltage regulator," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 31–44, Jan. 2016.

[2] M. Pons *et al.*, "Sub-threshold latch-based icyflex2 32-bit processor with wide supply range operation," in *Proc. 46th Eur. Solid-State Device Res. Conf. (ESSDERC)*, Sep. 2016, pp. 41–44.

[3] D. Bol *et al.*, "SleepWalker: A 25-MHz 0.4-V sub-mm$^2$ 7-$\mu$W/MHz microcontroller in 65-nm LP/GP CMOS for low-carbon wireless sensor nodes," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 20–32, Jan. 2013.

[4] H. Reyserhove and W. Dehaene, "A differential transmission gate design flow for minimum energy sub-10-pJ/Cycle ARM cortex-M0 MCUs," *IEEE J. Solid-State Circuits*, vol. 52, no. 7, pp. 1904–1914, Jul. 2017.

[5] W. Lim, I. Lee, D. Sylvester, and D. Blaauw, "Batteryless sub-nW cortex-M0+ processor with dynamic leakage-suppression logic," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2015, pp. 1–3.

[6] L. Lin, S. Jain, and M. Alioto, "A 595 pW 14 pJ/cycle microcontroller with dual-mode standard cells and self-startup for battery-indifferent distributed sensing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 44–46.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Workshop Mobile Cloud Comput. (MCC)*, New York, NY, USA, 2012, pp. 13–16. doi: 10.1145/2342509.2342513.

[8] S. Kim, J. P. Cerqueira, and M. Seok, "Near-Vt adaptive microprocessor and power-management-unit system based on direct error regulation," in *Proc. 43rd IEEE Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2017, pp. 163–166.

[9] J. Constantin, A. Bonetti, A. Teman, C. Müller, L. Schmid, and A. Burg, "DynOR: A 32-bit microprocessor in 28 nm FD-SOI with cycle-by-cycle dynamic clock adjustment," in *Proc. 42nd Eur. Solid-State Circuits Conf. (ESSCIRC)*, Sep. 2016, pp. 261–264.

[10] S. Paul *et al.*, "A sub-cm$^3$ energy-harvesting stacked wireless sensor node featuring a near-threshold voltage IA-32 microcontroller in 14-nm tri-gate CMOS for always-ON always-sensing applications," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 961–971, Apr. 2017.

[11] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[12] E. Flamand *et al.*, "GAP-8: A RISC-V SoC for AI at the edge of the IoT," in *Proc. IEEE 29th Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Jul. 2018, pp. 1–4.

[13] F. Conti *et al.*, "An IoT endpoint system-on-chip for secure and energy-efficient near-sensor analytics," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2481–2494, Sep. 2017.

[14] T. Karnik *et al.*, "A cm-scale self-powered intelligent and secure IoT edge mote featuring an ultra-low-power SoC in 14 nm tri-gate CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 46–48.

[15] D. Rossi *et al.*, "Energy-efficient near-threshold parallel computing: The PULPv2 cluster," *IEEE Micro*, vol. 37, no. 5, pp. 20–31, Sep./Oct. 2017.

[16] A. Pullini, D. Rossi, I. Loi, A. D. Mauro, and L. Benini, "Mr. Wolf: A 1 gflop/s energy-proportional parallel ultra low power SoC for IoT edge processing," in *Proc. IEEE 44th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2018, pp. 274–277.

[17] A. Waterman and K. Asanovic, "The RISC-V instruction set manual, volume I: BaseUser-level ISA," RISCV Found., Berkeley, CA, USA, Tech. Rep. UCB/EECS-2011-62, 2017.

[18] P. D. Schiavone *et al.*, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," in *Proc. 27th Int. Symp. Power Timing Modeling, Optim. Simulation*, Sep. 2017, pp. 1–8.

[19] M. Gautschi *et al.*, "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017.

[20] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 4114–4122. [Online]. Available: http://dl.acm.org/citation.cfm?id=3157382.3157557
[21] *The OpenMP API Specification for Parallel Programming*. Accessed: Jan. 30, 2019. [Online]. Available: https://www.openmp.org/
[22] P. Meinerzhagen, C. Roth, and A. Burg, "Towards generic low-power area-efficient standard cell based memory architectures," in *Proc. 53rd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 129–132.
[23] I. Loi, A. Capotondi, D. Rossi, A. Marongiu, and L. Benini, "The quest for energy-efficient I$ design in ultra-low-power clustered many-cores," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 2, pp. 99–112, Apr./Jun. 2018.
[24] W. S. Smith. (Jan. 1997). *Chapter 28: Digital Signal Processors*. [Online]. Available: http://www.dspguide.com/ch28/4.htm
[25] D. P. Subha, P. K. Joseph, R. U. Acharya, and C. M. Lim, "EEG signal analysis: A survey," *J. Med. Syst.*, vol. 34, no. 2, pp. 195–212, Apr. 2010. doi: 10.1007/s10916-008-9231-z.
[26] G. Frantz. (2004). *Comparing Fixed- and Floating-Point DSPs*. [Online]. Available: http://www.ti.com/jp/lit/wp/spry061/spry061.pdf
[27] F. Montagna, S. Benatti, and D. Rossi, "Flexible, scalable and energy efficient bio-signals processing on the pulp platform: A case study on seizure detection," *J. Low Power Electron. Appl.*, vol. 7, no. 2, p. 16, 2017. [Online]. Available: http://www.mdpi.com/2079-9268/7/2/16
[28] N. Ickes, Y. Sinangil, F. Pappalardo, E. Guidetti, and A. P. Chandrakasan, "A 10 pJ/cycle ultra-low-voltage 32-bit microprocessor system-on-chip," in *Proc. ESSCIRC*, Sep. 2011, pp. 159–162.
[29] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, New York, NY, USA, 2016, pp. 64–69. [Online]. Available: http://doi.acm.org/10.1145/2934583.2934624
[30] Y. Liu, W. Zhou, Q. Yuan, and S. Chen, "Automatic seizure detection using wavelet transform and SVM in long-term intracranial EEG," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 20, no. 6, pp. 749–755, Nov. 2012.
[31] M. Saint-Laurent *et al.*, "A 28 nm DSP powered by an on-chip LDO for high-performance and energy-efficient mobile applications," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 81–91, Jan. 2015.
[32] E. Beigné *et al.*, "A 460 MHz at 397 mV, 2.6 GHz at 1.3 V, 32 bits VLIW DSP embedding F MAX tracking," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 125–136, Jan. 2015.
[33] B. Zimmer *et al.*, "A RISC-V vector processor with simultaneous-switching switched-capacitor DC–DC converters in 28 nm FDSOI," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 930–942, Apr. 2016.
[34] B. Hübener, G. Sievers, T. Jungeblut, M. Porrmann, and U. Rückert, "CoreVA: A configurable resource-efficient VLIW processor architecture," in *Proc. 12th IEEE Int. Conf. Embedded Ubiquitous Comput.*, Aug. 2014, pp. 9–16.

**Antonio Pullini** received the M.S. degree in electrical engineering from the University of Bologna, Bologna, Italy, and the Ph.D. degree from the Integrated Systems Laboratory, ETH Zürich, Zürich, Switzerland.

He has been a Senior Design Engineer with iNoCs S.à.r.l., Lausanne, Switzerland, and he is currently with GreenWaves Technologies, Grenoble, France. His research interests include low-power digital design and networks on chip. In this field, he owns more than 50 papers in international peer-reviewed conferences and journals.

**Davide Rossi** received the Ph.D. degree from the University of Bologna, Bologna, Italy, in 2012.

He has been a Post-Doctoral Researcher with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi," University of Bologna, since 2015, where he is currently an Assistant Professor. His research interests focus on energy-efficient digital architectures. In this field, he has published more than 80 papers in international peer-reviewed conferences and journals.

**Igor Loi** received the Ph.D. degree from the University of Bologna, Bologna, Italy, in 2010.

He has been a Post-Doctoral Researcher with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi," University of Bologna, since 2006. He is currently with GreenWaves Technology, Crolles, France. His research activities are currently focused on ultralow power multi-core systems. In this field, he has published more than 40 papers in international peer-reviewed conferences and journals.

**Giuseppe Tagliavini** received the Ph.D. degree in electronic engineering from the University of Bologna, Bologna, Italy, in 2017.

He is currently a Post-Doctoral Researcher with the Department of Electrical, Electronic, and Information Engineering, University of Bologna. He has coauthored over 20 papers in international conferences and journals. His research interests include parallel programming models for embedded systems, run-time optimization for multicore and many-core accelerators, and design of software stacks for emerging computing architectures.

**Luca Benini** is currently the Chair of Digital Circuits and Systems at ETH Zürich, Zürich, Switzerland, and also a Full Professor with the University of Bologna, Bologna, Italy. He has published more than 800 papers, five books, and several book chapters. His research interests are in energy-efficient system design for embedded and high-performance computing.

Dr. Benini is a fellow of the ACM and a member of the Academia Europaea. He was a recipient of the 2016 IEEE CAS Mac Van Valkenburg Award.