Arnold: an eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End-Nodes

Pasquale Davide Schiavone, Davide Rossi *Member, IEEE*, Alfio Di Mauro, Frank Gürkaynak, Timothy Saxe, Mao Wang, Ket Chong Yap, Luca Benini *Fellow, IEEE*

Abstract—A wide range of Internet of Things (IoT) applications require powerful, energy-efficient and flexible end-nodes to acquire data from multiple sources, process and distill the sensed data through near-sensor data analytics algorithms, and transmit it wirelessly. This work presents *Arnold*: a 0.5 V to 0.8 V, 46.83 µW/MHz, 600 MOPS fully programmable RISC-V Microcontroller unit (MCU) fabricated in 22 nm Globalfoundries GF22FDX (GF22FDX) technology, coupled with a *state-of-the-art* (SoA) microcontroller to an embedded Field Programmable Gate Array (FPGA). We demonstrate the flexibility of the System-On-Chip (SoC) to tackle the challenges of many emerging IoT applications, such as *(i)* interfacing sensors and accelerators with non-standard interfaces, *(ii)* performing on-the-fly pre-processing tasks on data streamed from peripherals, and *(iii)* accelerating near-sensor analytics, encryption, and machine learning tasks. A unique feature of the proposed SoC is the exploitation of body-biasing to reduce leakage power of the embedded FPGA (eFPGA) fabric by up to 18× at 0.5 V, achieving SoA state bitstream-retentive sleep power for the eFPGA fabric, as low as 20.5 µW. The proposed SoC provides 3.4× better performance and 2.9× better energy efficiency than other fabricated heterogeneous re-configurable SoCs of the same class.

Index Terms—Embedded Systems, FPGA, Internet Of Things, Edge Computing, Microcontroller, RISC-V, Open-Source.

1 Introduction

The end-nodes of the IoT require energy-efficient, powerful, and flexible ultra-low-power computing platforms to deal with a wide range of near-sensor applications [1]. These SoCs must be able to connect to low-power sensors such as arrays of microphones [2], cameras [3], electrodes to monitor physiological activities [4], to analyze and compress data using advanced algorithms, and transmit them wirelessly over the network. Signal processing algorithms are executed in such devices to reduce complex raw data to simple classifications tags that classify data, to extract only relevant information (e.g., [5]), or to filter, encrypt, anonymize data. Compressing and distilling information that travels from IoT devices to the cloud, brings multiple benefits in power, performance, and bandwidth across the whole IoT infrastructure.

Depending on the constraints of the application such as flexibility, performance, power, and cost, IoT computing platforms can be implemented as hardwired Application specific integrated circuits (ASICs), programmable hardware (or soft-hardware) on FPGAs, or as software programmable on MCUs. Hardwired, fixed-function ASICs offer the best energy and energy efficiency, but they lack versatility and require long time-to-market [6]. Hence, their usage is preferred in highly standardized applications or specialized single-function products.

On the other side of the spectrum, MCUs are the de-facto standard platforms for IoT applications thanks to their high versatility, low-power, and low-cost. SoA MCUs can offer competitive Power-Performance-Area (PPA) figures by leveraging parallel Near-Threshold Computing (NTC) [7], and advanced low-power technologies such as Fully Depleted Silicon-On-Insulator (FDSOI) coupled with performance-power management techniques such as body-bias [8] and power-saving states [9]. As it has been shown in [9], [10], [11], [12], these techniques make possible the use of MCUs on edge computing devices, meeting PPA constraints for a wide range of applications in the IoT domain, yet providing high versatility. To increase performance, MCUs are often customized with on-chip full-custom accelerators that speed up the execution of part of the applications as for example neural-networks [13], frequency-domain-transforms [14], linear algebra [15], security engines [16]. The resulting heterogeneous system has thus both the flexibility of MCUs, and competitive performance and efficiency of hardwired ASICs on specific domains.

FPGAs fill the gap between ASICs and MCUs as they offer versatility via hardware programmability (which usually needs longer design and verification time than software), and they allow exploiting spatial computations typical of ASICs designs, as opposed to sequential execution. For these reasons, FPGAs are used in a wide range of applications, from machine learning [17], [18], [19], sorting [20], and cryptography accelerators for data centers [21], to smart instruments [22], analog-to-digital converters [23], to low-power systems for wearable applications [24], control-logic systems [25], and for implementing smart-peripherals connected to SoCs [26], [27].

Increased integration density of modern SoCs allowed a reasonably sized FPGA array to be integrated as part of an on-chip system. Such embedded FPGAs (eFPGAs) are used to enable post-silicon soft-hardware programmable functions in SoCs or MCUs to make updates on accelerators or custom peripherals. As for the FPGA case, hardwired accelerators or peripherals outperform their eFPGA-based implementations, but lack flexibility and post-fabrication reconfigurability. The benefit of integrating eFPGAs into SoCs is the possibility to increase performance by specializing the SoCs for one particular domain that can change over time, increasing the product life-time and application span.

In this paper, we present *Arnold*: a RISC-V based MCU extended with an eFPGA, implemented in GF22FDX tech-

P. D. Schiavone, A. Di Mauro, F. Gürkaynak, and L. Benini are with the Integrated Systems Laboratory, D-ITET, ETH Zürich, 8092 Zürich, Switzerland. D. Rossi is with the Energy-Efficient Embedded Systems Laboratory, DEI, University of Bologna, 40126 Bologna, Italy. T. Saxe, M. Wang, and K.C. Yap are with the QuickLogic Corporation, 2220 Lundy Ave, San Jose, CA 95131, United States of America.

nology. The contribution of the presented heterogeneous SoC design and silicon demonstrator are summarized as follows.

1) Architectural Flexibility: to enable architectural flexibility that fully exploits the configurable logic. The eFPGA is connected with the rest of the system with different interface options on the data-plane: *i*) a direct connection to the I/O DMA engine on the SoC - to process and filter data streams on their way from/to on-chip shared memory buffers in memory; *ii*) a high-bandwidth, low-latency interface to the memory of the RISC-V core - to interleave with zero-copy FPGA-accelerated parallel processing and sequential processing by the core; *iii*) a direct GPIO interface to implement master or slave peripheral ports for non-standard off-chip digital sensors or actuators. On the control plane we provide: *i*) an AMBA Advanced Peripheral Bus (APB) interface to allow the user to configure the mapped soft-hardware; *ii*) sixteen interrupts to notify the CPU.
2) Power Management: thanks to reverse body-bias (RBB) enabled by conventional-well FDSOI technology used for the physical implementation of the eFPGA fabric, leakage power can be reduced by 18x to 20.5 µW (featuring a fully state retentive bitstream) when eFPGA functionality is not required.
3) Leading Edge Performance and Energy Efficiency: the SoC achieves SoA performance and efficiency, leveraging a voltage and frequency scalable architecture from 0.5 V to 0.8 V, with a peak energy efficiency of 46.83 µW/MHz at 0.52 V and a maximum frequency of 600 MHz at 0.8 V. The proposed SoC achieves 3.4× better performance and 2.9× better energy efficiency than SoA MCUs augmented with eFPGA built for the same power target applications [28], [29], [30].

The remainder of the paper is organized as follows: Section II provides a review of related works. In Section III, the architecture of the proposed SoC is described, including all its components. In Section IV and V, the software and tools for the proposed SoC, its physical design, and silicon measurements are described respectively, whereas, in Section VI, use cases for the proposed work are reported as application examples. The paper concludes in Section VII.

2 Related Work

In this section, we review devices that define the boundaries of its design space: MCUs, FPGAs, eFPGAs, and heterogeneous reconfigurable SoCs.

2.1 MCUs

In the context of edge-computing systems, MCUs need to provide significant performance within a limited power budget, and the flexibility needed to cope with a wide variety of sensors and algorithms. Most Off-the-Shelf (OTS) MCUs use energy-efficient Central Processing Units (CPUs) based on ARM Cortex-M family of cores, such as the NXP i.MXRT1050 [31], the STMicroelectronics STM32L476xx family [32], or the Silicon Labs EFM32 Giant Gecko 11 [42], all featuring a power budget within a few tens of mW. To interface with a large variety of external devices, these systems offer a wide set of peripherals such as I2C, UART, SPI, and GPIOs. SoAs energy-efficient MCUs optimized for ultra-low-power (3µW/MHz) [12] and performance (938 MHz) [33] have been implemented in FDSOI technology leveraging body-biasing to compensate process-voltage-temperature (PVT) variations, and to control performance and power to achieve higher energy efficiency.

Although software provides high versatility, some applications still need performance that a single CPU cannot deliver. For this reason, several MCUs are extended with custom accelerators, for example, the binary neural-network accelerator presented in [13] or the cryptography engine integrated into [42]. To improve flexibility with respect to dedicated accelerators, there are MCUs that combine multiple heterogeneous CPUs managing different tasks, for example, the NXP i.MX 7ULP Applications Processor [40], which combines an application ARM processor (ARM Cortex-A7) with a real-time CPU (ARM Cortex-M4) for performance and power trades off. Other approaches leverage parallel clusters of processors to improve the energy efficiency of near sensor analytics workloads, such as Mr.Wolf [9], featuring an 8-core cluster based on DSP-enhanced RISC-V cores controlled by a smaller core managing the I/Os, the runtime, and SoC control functions. These systems can choose to divide the workload as a subset of processors to meet the performance target at the lowest energy budget [50]. Finally, heterogeneous systems like GAP-8 from GreenWaves Technologies [45] and Fulmine [46], combine both custom and parallel software programmable accelerators providing a step forward for performance and flexibility of embedded platforms for signal processing. Although these platforms are compelling and flexible to run signal processing tasks for typical end-nodes, they are less efficient than reconfigurable devices such as FPGAs when dealing with non-standard sensors.

2.2 FPGAs

FPGAs are reconfigurable devices that on one hand can exploit spatial computations typical of ASIC designs but still retain the capability of being reconfigured after fabrication. They range from high-end FPGAs used for acceleration of high-performance workloads to ultra-low-power, small and low-cost technology implementations, as discussed further in this section.

High-end FPGAs, such as the Xilinx Virtex Ultrascale devices [43] and the Intel Cyclone 10 GX device [44], have millions of LUTs, flip-flops, DSP-blocks, and SRAM macros containing Mbytes of memory. To extended their capabilities in the embedded application domain running software, such FPGAs are often programmed with soft-CPUs [51]. The users can implement a deeply pipelined core with multiple issues to achieve high performance, or a tiny soft-core with a small area footprint for control applications [52], [53], and offload part of the control functionalities executed in SW to the soft-CPU. For example, Choi et al. [54] presented a FPGA-based 20 k-Word speech recognizer using a Xilinx Virtex-4 FPGA where the computationally less demanding tasks are executed in SW, whereas the rest of the algorithms is accelerated in HW.

As soft-cores are limited in performance [55] and occupy resources, FPGAs are often extended with hard-CPUs as

Table 1
Summary of related work: (left) MCUs programmable via Software and their accelerators; (center) FPGAs programmable via Soft-Hardware design; (right) eFPGAs programmable via Soft-Hardware design.

MCU		FPGA		eFPGA	
Single Core	[12], [31], [32], [33]	**Low Power**	[34], [35]	**StandAlone**	[36], [37], [38], [39]
SW Accelerator	[9], [40]	**Low Power SoC**	[41]	**MCU SoC**	[28], [29], [30],
HW Accelerator	[13], [42]	**HP**	[43], [44]		**This Work**
HW/SW Accelerator	[45], [46]	**HP SoC**	[47], [48]	**HP SoC**	[49]

application processors (usually ARM-based embedded processors such as the Xilinx Zynq-7000 SoC [47] and Intel Arria V SoC [48], in the case of Microsemi PolarFire [56] RISC-V processors). As a result, high-end FPGAs have typical power consumption in the order of tens of Watts [57], and they are usually used as high-performance accelerators on servers connected via Ethernet or PCI interfaces [58].

In the low-power domain, FPGAs are typically realized with a less aggressive process than high-end FPGAs. They are usually smaller, cheaper, and as a result, have lower performance than the others. Examples are the Microsemi IGLOO nano [34], which has up to 3 k logic elements[1], or the Lattice Semiconductor iCE40 UltraLite [35], which has more than 1K of LUTs+flip-flops. Both consume from a few µW to hundreds of mW. These FPGAs are used to extend the I/O subsystem of embedded controllers [59], even with simple data pre-processing engines to lower the bandwidth coming from sensors [24], [60]. In the low-end space, FPGAs can also be extended with CPUs to leverage HW/SW co-designed IoT nodes. An industrial RISC-V based soft-core is provided by the Microsemi Mi-V RV32, ready to be integrated into the SmartFusion2 SoC [41] or in the IGLOO FPGA [61] in an area footprint of 10 k-26 k LEs. Other RISC-V based solutions have emerged during the *RISC-V SoftCPU Contest* in December 2018, with the VexRiscv soft-core as the winner. Hard-CPUs are also used as in the Microsemi SmartFusion2 SoC in 65nm [41], which proposes an MCU-class (ARM Cortex-M) core running at 166 MHz and an FPGA with DSP blocks and up to 150 k logic elements, 656 kB[2] of memory, and power consumption in the order of hundreds of mWatts. Examples that use the Microsemi SmartFusion2 SoC can be found in Gomes at al. [62], which proposes a system where most of the tasks are executed by the ARM core, whereas the FPGA is used for accelerating critical network kernels. In Fournaris et al. [63], the operating system, and user interfaces run in software, whereas the FPGA is used to collect sensor data, extract features, and to calculate the nearest neighbor on the extracted information. The system runs at 160 MHz consumes 4.96 mW on the CPU part and 153.97 mW on the FPGA side. While their power consumption is within range of IoT applications, these FPGAs are limited in performance and thus not suitable for computationally intensive applications. To enrich the functionalities of deeply embedded SoCs, FPGA vendors started to develop and commercialize FPGA IPs that can be integrated into SoCs, presented in the following section.

1. One logic element is composed of one 4-input LUT and one flip-flop
2. 512 Bytes of Non-Volatile Memory

2.3 eFPGAs

eFPGAs are FPGA IP cores specifically meant to be integrated into SoCs to extend them with programmable logic. Unlike the FPGAs presented in the previous section, eFPGAs are not meant to be used standalone, but are designed with the goal of enhancing the capabilities of the SoCs. Vendors provide tools to allow eFPGAs to be customized to the SoCs and properties like the number of arrays, with a given number of LUTs, DSP blocks, flip-flops, I/O pins, etc. can be configured. eFPGAs can be provided as soft-IP [30], [36], described in RTL and synthesized with the rest of the system, or hard-IP [28], [29], [49] as hard-macros with pre-determined physical layout, featuring a different trade-off between performance and cost. Although soft eFPGA macros are easily portable from different technology nodes as they are made by standard cells, hard-macro eFPGAs, which are usually custom-designed at layout level, feature significantly better PPA figures.

For example, in Renzini et al. [30], a soft-IP is complementing a MCU for power control applications is implemented using a 90 nm Bipolar CMOS DMOS (BCD) technology. This eFPGA is relatively small (only 96 4-input LUTs and 192 flip-flops) and connected exclusively to the I/O subsystem to implement low-latency and flexible control tasks such as Pulse Width Modulation (PWM). Several companies are providing hard-IP blocks, as Achronix [37], which provides 7nm FinFET eFPGAs, Flex-Logix [38], which provides from 12 nm to 180 nm eFPGAs macros, QuickLogic Corporation [39], which provides from 22 nm to 65 nm core IPs, and Menta [36], which provides IPs from 10 nm to 90 nm. Several heterogeneous reconfigurable SoCs have been presented in the last years, ranging from high-performance systems to low-power embedded systems. Whatmough et al. presented a 25 mm^2 SoC implemented in 16 nm FinFET technology featuring two ARM A53 cores, a quad-core datapath accelerator, 4 MBytes on-chip SRAM, and a 2×2 FlexLogic eFPGA macro featuring hardwired DSP slices [49]. The proposed SoC can achieve up to 28.9× better energy-efficiency when DSP and crypto algorithms are executed on the eFPGA rather than the ARM cores.

In the embedded domain, several solutions have been proposed in different technology nodes. Borgatti et al. [28] implemented a 180 nm 20mm^2 SoC, where eFPGA is integrated with the CPU pipeline to implement a reconfigurable Application Specific Instruction Processor (ASIP) SoC, with the eFPGA implementing custom instructions. In addition, the eFPGA is connected to the system bus and I/O pads. The system reports up to 10× performance gain using instruction extensions to accelerate face-recognition algorithms and 2× for I/O intensive tasks when dealing

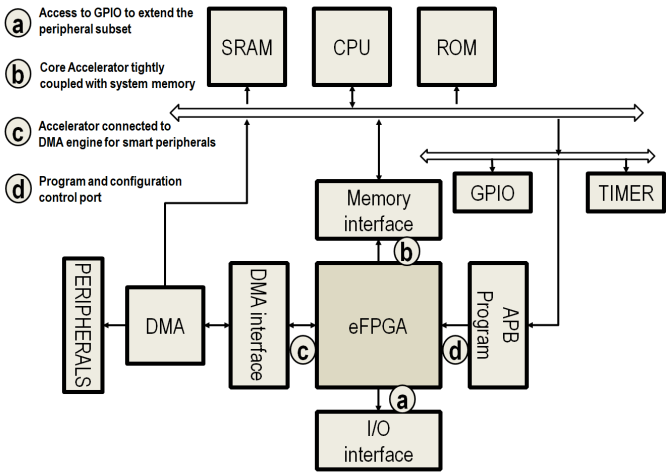


Figure 1. MCU-eFPGA SoC architecture. eFPGA connections towards the MCU and to the external peripherals are highlighted.

with camera peripherals with pre-processing. Lodi et al. [29] implemented a 42 mm^2 SoC in 130 nm, where the CPU pipeline is directly connected with the eFPGA to implement custom instructions, whereas a second eFPGA is connected to the system bus and I/O pads. The system reports up to 15× performance gain and 89% energy saving by exploiting the eFPGAs to accelerate a set of data processing algorithms. However, as a consequence of using a mature technology node, the eFPGAs (~15 kGE) presented in the proposed SoCs feature limited capabilities and performance.

To boost signal processing workloads, both hard and soft eFPGAs can have digital signal processor (DSP)-blocks included in the IP itself, or they can have pins dedicated to communicating with external blocks, featuring, once again, a different trade-off between time to market for DSP-blocks customization at design time. The first ones can be used by eFPGA synthesis tools to map user-designs in DSP-blocks implicitly, whereas in the second case, the user explicitly designs logic in the eFPGA to interact with the external blocks. All the works featuring DSP-blocks so far belong to the first category, whereas the proposed work has MAC-blocks external to the IP macro.

In this work, we propose an SoC featuring an advanced microcontroller augmented by an embedded eFPGA for IoT applications in 22 nm process technology. Differently from what has been proposed in Whatmough et al. [49], we target a much lower power budget. The proposed SoC utilizes the eFPGA to enhance the I/O capabilities of the SoC, by performing I/O pre-processing tasks as well as being used as a tightly coupled accelerator. The proposed solution provides 3.4× better performance and 2.9× better efficiency than state-of-the-art heterogeneous reconfigurable SoCs. One key feature of the SoC is the unique capability to exploit reverse body biasing enabled by FD-SOI technology to implement a 20.5 µW state-retentive deep-sleep mode for the eFPGA. This point is further discussed in Section 5.

3 Arnold Architecture

The proposed system is built around an in-order RISC-V core[3] based on [64], optimized for signal processing, featuring a 4-stage pipeline, and achieving 3.19 Coremark/MHz and up to 2.4 eight-bit GMAC/s (at 600 MHz). The core implements the RISC-V 32 bit integer (I), multiplication and division (M), single-precision floating-point (F), and compressed (C) Instruction Set Architecture (ISA) extensions (RV32IMFC) [65]. In addition, the core has been extended with custom instructions to speed up data processing applications such as zero-overhead hardware loops, automatic increment load/store instructions, bit manipulations, and packed-single-instruction-multiple-data (pSIMD) operations between vectors of 4 bytes or 2 half-words at a time.

To protect sensitive parts of the system from corrupted user applications, we extended the CPU with a RISC-V compliant Physical Memory Protection (PMP) unit that can control read, write, and execute permissions on regions of the physical memory. The implemented RISC-V PMP supports all address matching schemes as: naturally aligned power of 2 regions *NAPOT* (including 4 bytes alignment *NA4*); and the top boundary of an arbitrary range *TOR*. The PMP occupies only 14% of the total CPU area due to the extra registers and comparators needed to implement the specifications and provides much-needed security features for user-applications in the IoT domain. In the proposed SoC, the CPU is responsible for executing the runtime to manage the system and to execute user applications to process data or to control external peripherals, as well as to configure and control the eFPGA itself.

3.1 Memory Subsystem

The memory system, composed of 512 kB of static random-access-memory (SRAM), is shared among the CPU (instruction and data), the I/O DMA (μDMA) (RX and TX), the JTAG, and the eFPGA masters. The memories are slaves of the system bus, which is based on a single-cycle latency logarithmic interconnect [66] (XBAR bus in Fig. 2). In case two or more masters request to access the same slave, a round-robin arbiter selects the master that first communicates with the slave to solve the conflict. The shared memory consists of four word-level interleaved memory banks, each with 112 kB each, and two memory banks of 32 kB featuring a non-interleaved address scheme. Every memory bank is a composition of single-port 4096 by 32 bit words (16 kB) memory cuts optimized for density and power. The size chosen for the memory cuts allows to place them comfortably during the physical implementation as described below, and concurrently to meet the frequency target.

The chosen interleaving scheme for the four 112 kB (448 kB) memory portion approximates a multi-port memory access, and it increases the bandwidth up to 4× when multiple masters are loading or storing data sequentially, which is the typical case for most DSP applications. When low-latency single-cycle accesses with no contention are needed, the two private banks can be used, which offer a bandwidth of 19.2 Gbps each. In the proposed MCU, they

3. The OpenHW Group CV32E40P is freely downloadable at https://github.com/openhwgroup/ under the SolderPad license.

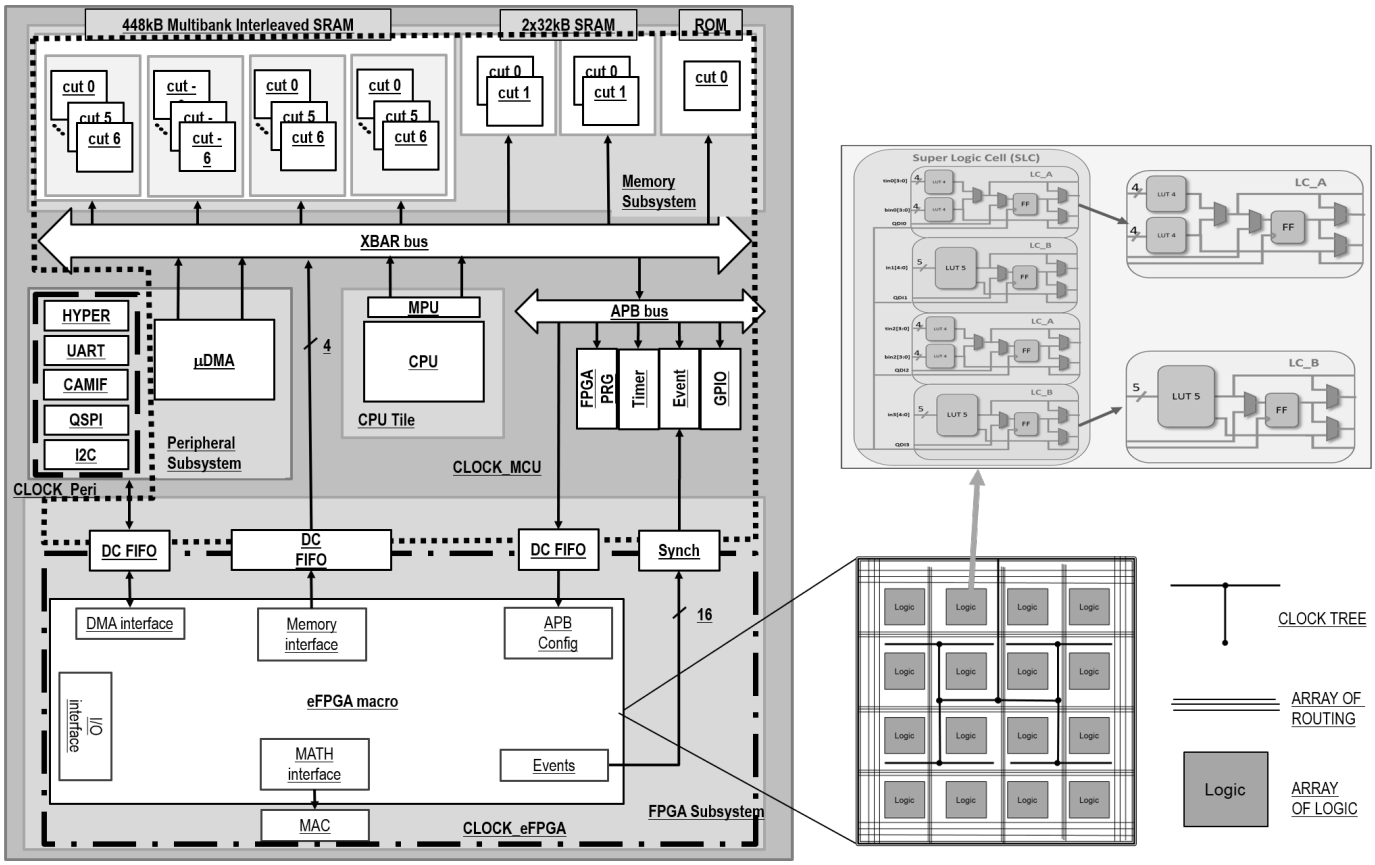


Figure 2. Detailed block diagram of the proposed design. The eFPGA (bottom) connected with the MCU and its private MAC units in a clock domain (CLOCK eFPGA). Peripherals (center – left) are directly connected to the µDMA in the Peripheral subsystem and operate on the CLOCK Peri clock domain. The rest of the system works in the CLOCK MCU domain. The CPU runs the SW and orchestrates the whole system.

are used to store private CPU data such as the stack and instruction binary. In this way, the interleaved part can be used by the other masters with no conflicts. This solution avoids the use of power and area hungry multi-port memory cuts, still providing low-latency access to memory, increasing the total energy efficiency. A read-only-memory (ROM) has also been implemented to store the boot instructions responsible for setting the system upon reset.

3.2 I/O subsystem

The I/O subsystem is composed of a broad set of peripherals that include JTAG, HyperRam, UART, Camera Interface, quad-SPI, and I2C, which communicate with the shared memory system through an autonomous µDMA based on [67]. The µDMA is a smart-engine that allows peripherals to control transfers to/from memory without the need for the CPU continuous control. The HyperRam peripheral is particularly interesting as it allows to access off-chip memory with a bandwidth of 800 Mbps, extending the MCU with larger memory capacity, useful for holding several eFPGA bitstreams.

The µDMA has two ports towards the main memory, one to transmit and one to receive data from peripherals. At 600 MHz, the µDMA has an aggregated bandwidth equal to 38.4 Mbps. Except for the JTAG, which is directly connected to a master port of the system bus, the other peripherals are controlled by the µDMA core, which handles memory requests in a time-multiplexed fashion. The µDMA control registers are used to select the active peripheral, the peripheral clock frequency, number of transfers, etc. Other peripherals, such as SoC control registers, timers, GPIOs, and event units are also included in the proposed MCU and accessible through the APB bus.

3.3 Clock subsystem

Arnold includes three Frequency-locked loops (FLLs) that take as input an external 32 kHz reference clock and provide internal clocks up to 2.1 GHz. One FLL each is used to provide the clock to the eFPGA, the peripheral subsystem and the remaining modules as CPU, memories, busses, etc. The eFPGA has access to six clock sources: four from external GPIOs; one from the eFPGA FLL block; and one from an integer frequency divider from the same FLL.

3.4 eFPGA subsystem

The eFPGA is tightly coupled to the system to minimize the overhead of communications with the CPU. It has 3712 pins to be used to connect the IP with the rest of the SoC. In this work, we designed a novel, highly flexible 4-mode SoC interface to:

(a) an I/O interface with direct connections toward the pad frame of the system, enabling the implementation of custom off-chip interfaces;
(b) a memory interface suitable for shared-memory accelerators implemented on the FPGA logic and tightly coupled with the CPU;

(c) an I/O DMA interface suitable for implementing I/O filtering functions for data streamed into the system from the standard I/O;
(d) an APB configuration and control interface suitable for controlling the programmable logic.

The I/O interface is made of 41 sets of three signals (input, output, direction) from the eFPGA to the GPIOs. This interface is used for custom I/O protocols, which are challenging to implement efficiently in SW due to latency constraints. Each I/O pad can be either used by a peripheral (quad-SPI, Camera Interface, etc.), or by software (Core GPIO), or by the eFPGA. Multiplexers controlled by SoC registers drive the functionality mode of each pad.

The memory interface implements the protocol presented in [66]. The proposed SoC has four interfaces connected as master ports in the bus, providing up to 128 bit memory operations (load or store) per transaction. Access to the on-chip SRAM is provided through four 32 bit 4 words dual-clock FIFOs to allow the MCU and the eFPGA subsystem to operate at independent frequencies. This is a crucial feature since the eFPGA usually runs at a lower frequency than the rest of the SoC and its frequency depends on the user design. For security reasons, the eFPGA memory interface has only access to SRAM banks and not to APB peripherals and boot ROM.

The I/O DMA interface is composed of one receive (RX), and one transmit (TX) bus featuring a ready/valid handshaking, plus one 32 bit configuration bus as described in [67]. The configuration bus allows controlling the peripherals mapped into the eFPGA with external registers which can avoid the use of the APB interface described below, and thus save resources. In addition, this interface can be used to stream data through the µDMA without using eFPGA resources for the address generation logic as it would with the memory interface. In this case, the µDMA transfers data from the eFPGA to memory (and vice versa) linearly. Communication between the µDMA and the eFPGA happens using two 32 bit 4 words dual-clock FIFOs.

Designs mapped into the eFPGA (as accelerators or peripherals) can be controlled by registers through the APB configuration and control interface. Such an interface is made of a 7 bit address, 32 bit data read, and data write, write-enable, ready, peripheral select and enable signals (75 pins). One 32 bit 4 words dual-clock FIFO is used for communications between the MCU and the eFPGA.

In addition to the four interfaces mentioned above, the eFPGA can generate sixteen events to interact asynchronously with the CPU, avoiding inefficient polling operations and saving power. In fact, the eFPGA event pins are connected to dual-clock event-propagators that notify the events to the CPU as dedicated interrupts requests. The interrupt service routines are user-defined, and they can be used to handle the eFPGA requests, for example, starting a new I/O transaction, or programming the new acquired data pointers to start processing them in case of accelerator design.

To improve computational arithmetic density, two synthesizable parallel-vectorial Multiply-and-Accumulate (MAC) accelerators are connected to the eFPGA to compute four 8 bit, two 16 bit, or one 32 bit MAC operations for each unit. The two MAC blocks are connected via 310 pins each, which control the MAC blocks, whether data comes from the eFPGA or the MAC buffers, the input and output data, and the vector mode (8, 16, or 32).

The CPU programs the eFPGA through another APB interface. Such master interface is connected to the eFPGA Fabric Configuration Block (FCB), which is responsible for controlling the eFPGA, managing the power procedures, and report the actual status of the eFPGA. The eFPGA binary is 225.5 kB, small enough to be contained in the on-chip SRAM. To program the macro, the CPU reads the binary from an external memory to the on-chip memory, then the CPU reads the binary array and writes its content to the APB FCB via non-critical load and store instructions.

The eFPGA fabric is organized in four quadrants with dynamic reconfiguration capabilities, each one composed of an array of 16x16 Super Logic Cells (SLCs). Each SLC has four logic cells that are organized in two sub-logic clusters: two instances of logic cell A (LCA) and two instances of logic cell B (LCB), as shown in Fig. 2. Both LCA and LCB also include one register and multiple multiplexers that enable the logic cell to perform different functions (e.g., combinatorial, sequential, or both). If a logic cluster or a highway network within the SLC is not used, it is powered off to save static power. A shared register clock, set, and reset signals for all four logic cells helps reduce routing congestion. If the logic cluster or highway network within the SLC is not used, it is powered off to save static power.

4 eFPGA Software and Tools

To use the eFPGA in the *Arnold* SoC, the user writes HDL code (VHDL, Verilog or SystemVerilog) and synthesizes it with Mentor Graphics Corporation © Precision RTL Synthesis OEM Quicklogic tool. The synthesized design is then placed and routed with the QuickLogic Aurora Software Tool Suite (Aurora). The user must map each of the soft-module interface pins to the corresponding pin of the eFPGA hard-macro. For example, the user may define the memory interface request signal as "MemREQ_output", in the Aurora tool, the user may specify that the signal is connected to the 3rd memory interface of the eFPGA specifying that "MemREQ_output" is connected to "tcdm_req_p3_o" pin. The eFPGA pin has been assigned to its interface functionality at SoC design time to optimize the place and route phase.

Once the constraints and the pin mapping have been defined, Aurora performs logic optimization on the synthesized design, places, and routes it. It also generates static timing analysis and the bitstream containing the binary of the user-design. The binary is then loaded into the main memory by the CPU. The CPU stores each binary word into the bitstream registers. Once the eFPGA has been programmed, the CPU can control the design with user-defined registers mapped into the eFPGA APB interface described above to start the design, to check the status, etc. Application Programming Interfaces (APIs) have been developed to provide C procedures for the user. In particular, functions to RESET the eFPGA, to load the bitstream, and to wait for the end of the eFPGA computation (*wait_fpga_eoc*) have been implemented for fast integration into the user application. The *wait_fpga_eoc* routine leverages the "wait

Table 2
Area distribution of the main components of Arnold.

Module	Area [µm^2]	Percentage
CPU	27'186	0.54%
Main Memory	734'232	14.46%
I/O DMA	21'755	0.43%
eFPGA subsystem	63'946	1.26%
PAD Frame	229'519	4.52%
eFPGA Macro	4'000'000	78.79%

for interrupt" (WFI) RISC-V instruction to clock-gate the CPU to save dynamic power.

5 Arnold Physical Design

The proposed SoC fabricated in GF22FDX 10 Metal technology occupies 3×3 mm^2. FDSOI technology has been chosen as it provides performance and power knobs through body-biasing, and it is highly energy-efficient over a wide Vdd range [68] as confirmed by our results discussed in the Subsection 5.1. The synthesis tool used for this project is Synopsys © Design Compiler 2017.09, whereas the place and route tool used is Cadence © Innovus 18.11. The design has been closed at 430 MHz for the MCU side and for up to 100 MHz for the eFPGA soft-designs. Worst-case conditions at 0.72 V for setup constraints, and best-case conditions at 0.88 V for hold constraints between -40°C and 125°C have been used to guarantee performance across the process, voltage, and temperature variations.

The die picture and floorplan of the chip are shown in Fig. 3. The eFPGA macro is 2×2 mm^2, and it has been placed in the bottom left of the design. The memory cuts have been placed to the right of the eFPGA. The eFPGA memory interface pins have been assigned to the right part of the eFPGA to minimize routing efforts and to minimize the congestion issue as the path towards the memory is the most critical. The core has also been automatically placed close to the memory to minimize timing penalties. The eFPGA pins for the MAC blocks accelerators have been placed to the top part, where the local math accelerator SRAM buffers have been placed. On the left part of the eFPGA, the pins towards the µDMA, the user APB interface, and the 16 events pins have been assigned. GPIOs pins are spread along the four sides of the eFPGA. The six clock pins of the eFPGA are located three on the top and three on the bottom side. The three FLLs have been placed on the top part of the chip, whereas the standard cells have been automatically placed by the place and route tool.

The effective area occupied by the chip is 5.11mm^2, of which the eFPGA macro occupies 78% (4mm^2) and the MCU 22% (1.11mm^2). The main memory occupies 14.46% of the system area, whereas the I/O subsystem and the CPU take only 0.43% and 0.54%, respectively. The eFPGA subsystem components occupy 1.26% of MCU area. The eFPGA subsystem is a set of modules that interact directly with the eFPGA macro, dual-clock FIFOs, the FCB, the MAC accelerators (including memory buffers), and clock multiplexing logic. Table 2 shows the area distribution of the chip.

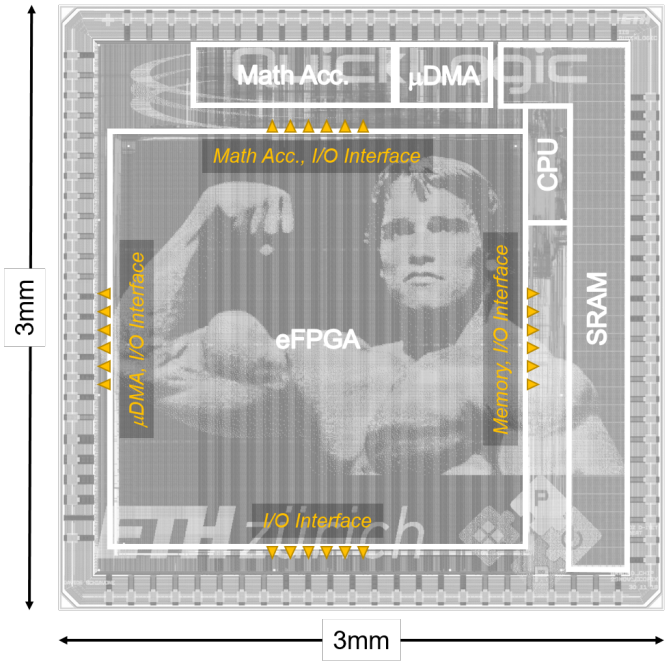


Figure 3. Die photo of the proposed design with the main components and eFPGA pins highlighted.

The MCU and the eFPGA operate at the same supply voltage, but the eFPGA can be switched off from external power managers. The range of operation is between 0.5 V to 0.8 V. To reduce the leakage power while preserving the eFPGA configuration during state-retentive deep sleep states, RBB is applied from an external generator to minimize on-chip implementations overheads. On the other hand, forward body-bias (FBB) is applied to the CPU, memory, and the rest of the logic to increase performance [33], [69].

5.1 Performance and Energy Efficiency

In this subsection, measured results at room temperature from the implemented chip are reported and discussed. Performance and power results have been measured using an Advantest SoC V93000 ASIC tester. Fig. 4 (left) shows the maximum frequency (a), power consumption (b), and power density (c) of the MCU during the execution of a matrix multiplication at different supply voltages. Measured results at ambient temperature show a maximum frequency of 135 MHz and power consumption 11.88 µW/MHz at 0.49 V, up to a maximum of 600 MHz at the nominal 0.8 V while consuming 26.18 µW/MHz. The maximum frequency at 0.49 V is comparable with commercial single-core MCUs performance while achieving very low power consumption thanks to voltage scaling. When high performance is needed, 600 MOPS can be achieved at a maximum power consumption of 16 mW. The leakage power of the whole MCU ranges from 0.53 mW (33%) to 2.39 mW (15%) at 0.49 V and 0.8 V respectively. Fig. 4(g) shows the effect of the FBB on the MCU power consumption, and Fig. 4(h) on the frequency. The MCU can run up to 20% faster at 0.6 V at the price of 43% higher power consumption, whereas the effect of FBB is smaller when applied at 0.8 V (only 5% faster) for a maximum frequency of 630 MHz. The effect of

the magnified impact of body biasing at low voltage is a well-known effect seen in near-threshold FD-SOI chips [70].

Fig. 4 (center) shows the eFPGA measured results. Fig. 4(d) shows the maximum frequency of two different designs: *FF2SOC* is an eight-way parallel 32 bit accumulator that reads values from the SoC memory and accumulates them in eight different registers. The signature can be read with the APB interface; *FF2FF* is a nine bit counter that divides the eFPGA clock by 512 and drives a GPIO with the divided clock. The designs are different as the *FF2SOC* communicates with synchronous elements in the SoC (dual-clock FIFOs), and thus its maximum frequency is bounded by the internal delays of the eFPGA and the logic outside its boundary, whereas *FF2FF* has been designed to measure only the flip-flop to flip-flop delay, without taking into account the propagation and setup timing of the eFPGA and the external logic at its boundary. The output of the Q-pin of the MSB flip-flop of the nine bit counter is directly connected to the GPIO, and the frequency is measured with an oscilloscope. From measurements we determined a maximum frequency of 475 MHz at 0.8 V and 260 MHz at 0.65 V. *FF2SOC* occupies 15% of the internal eFPGA resources and it can run from 26.38 MHz, consuming 34.34 µW/MHz at 0.52 V, to 126.88 MHz at 0.8 V consuming 47.98 µW/MHz (Fig. 4(e)).

The eFPGA *FF2SOC* leakage power is 0.38 mW at 0.5 V, up to 2.18 mW at 0.8 V. The power has been measured separately from the rest of the system as the power grid stripes of the eFPGA are different from the MCU ones. The power overhead added by the eFPGA is affordable in the IoT domain, making the integration of such programmable arrays a viable option for the next generation of edge-computing nodes. The eFPGA leakage power consumption is reduced via state-retentive deep sleep states applying RBB, resulting in a minimum leakage power of 20.5 µW at 0.5 V and 374.2 µW at 0.8 V and 1.8 V reverse body-bias as shown in Fig. 4(i), i.e., a 5.8×(at 0.8 V) to 18× (at 0.5 V) reduction can be achieved thanks to RBB. This result makes the eFPGA power consumption significantly reduced when not used, minimizing the integration cost and overhead. Fig. 4(f) shows how the power consumption changes with respect to the utilization rate. A design with a parametrizable number of adders has been implemented in the eFPGA to measure the power consumption with respect to the utilization rate. When running at 80 MHz, 0.75 V, results show an energy-efficiency of 0.40 µW/MHz/SLC, being leakage dominated when <20% of resources are utilized. The best energy-efficient point of the whole system is 46.83 µW/MHz (eFPGA consumes 28% of total power) achieved in near-threshold at 0.52 V, when the core and the eFPGA are running at 183.6 MHz and 26.38 MHz respectively. This result has been measured when the eight parallel 32 bit accumulators are mapped on the eFPGA.

6 Use Cases

To demonstrate the flexibility and efficiency of our heterogeneous reconfigurable SoC, three different use cases have been implemented, highlighting the versatility offered by embedded programmable logic.

6.1 I/O subsystem accelerator

In the context of applications for bio-signal processing, it is common to extract features in the frequency domain to classify activities sensed from skeletal muscles or the brain [72]. Wavelet or Fourier transforms are used to convert the signal from the time to the frequency domain, then features like the spectral power, are extracted and used by a pattern recognition algorithm. For this reason, a peripheral that extracts relevant information of the signal acquired from the sensors has been developed and mapped to the eFPGA to alleviate the pre-processing part of the CPU, which then classifies the activity starting from the extracted features. The peripheral accelerator mapped on the eFPGA consists of an SPI module extended with computational capabilities to calculate the Haar Discrete Wavelet (HDWT), which is an attractive algorithm to implement in an eFPGA as it does not require multipliers [73].

The accelerator is configured to acquire N samples of 16 bit of raw data coming from ADCs, and to store the Approximated and Detailed Wavelet Transform coefficients in the main memory. Also, coefficients can be stored in an 8 bit format to compress information in the main memory. The accelerator is programmed at the beginning with the number of samples to acquire and the output vector pointers. The eFPGA autonomously loops over SPI transactions and stores to the main memory, either the raw data or the Approximated and Detailed coefficients of the HDWT. When all the N data have been stored into the memory, an interrupt notifies the core at the end of the acquisition.

Moreover, a second function has been mapped to the custom SPI peripheral, namely, to extract 4 bits local binary patterns from a stream of data coming from sensors, as an algorithmic approach presented in [74]. In this case, for each data acquired, the eFPGA reuses the subtractor instantiated for the HDWT to compare the last two samples. If the last sample is greater than the previous one, it stores 1 in a 4 bit shift register, otherwise 0. The accelerator stores into memory a 16 bit value every four samples, each representing four single sample overlapping windows. The core takes 8 cycles for each tuple approximate-detail coefficient to compute the HDWT, whereas it takes 16 cycles for the local binary pattern. The eFPGA instead computes the features during the acquisition of the signal from SPI without adding latency overheads.

The design utilizes 20% of the available SLCs, and it uses a memory interface port, the APB interface, four GPIOs (3 output pins and 1 input pin), and it generates one event.

6.2 Custom I/O interface

IoT devices are often connected to custom peripherals that need more control pins that the usual peripherals as SPI, UART, I2C, I2S, etc. In this case, off-chip FPGAs are selected to implement the control part of the custom peripheral on one side and to communicate with the MCU with a standard protocol (e.g., SPI) to the other side. An example of a custom peripheral is a neuromorphic vision sensor [75] or event-based audition sensors [76]. Another example where FPGAs are used to control and transfer data are bridges for off-chip accelerators, for example, [77], or [78]. In this context, to illustrate the flexibility of the MCU+eFPGA combination, a

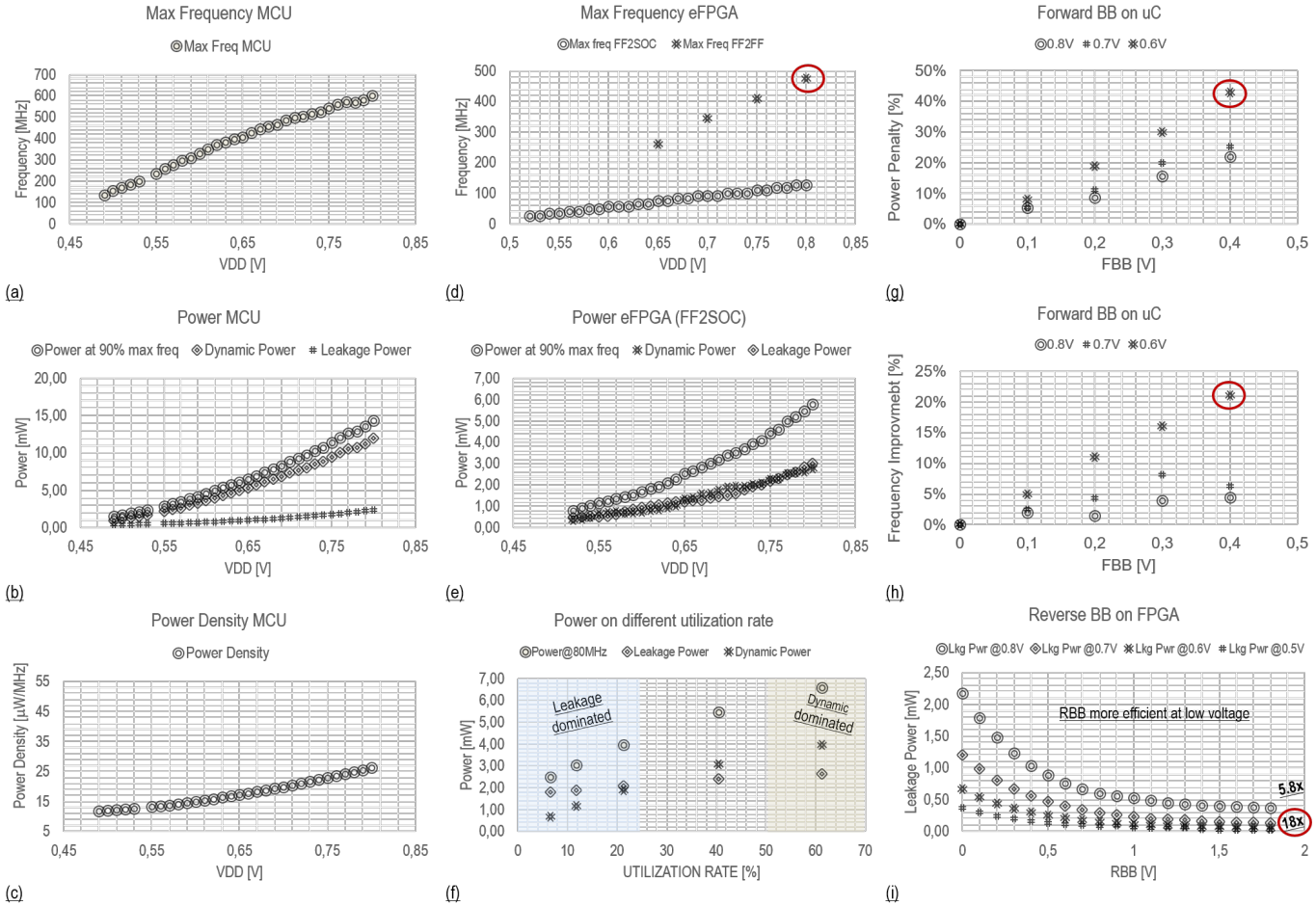


Figure 4. Frequency (a), power consumption (b), and energy-efficiency (c) with respect to the supply voltage of the MCU part of the proposed design. In the center, frequency (d) and power of the eFPGA macro with respect to the supply voltage (e) and power with respect to the utilization rate (f). The effect of the FBB on power (g) and frequency (h) on the MCU. The effect of RBB on the eFPGA leakage power during state-retentive deep-sleep mode (i).

controller for the systolic Long short-term memory Recurrent Neural Network (LSTM-RNN) accelerator presented in [77] has been implemented in the eFPGA. The LSTM-RNN accelerator is made of four chips implemented in UMCL 65 nm technology, and it is used to classify phonemes in real-time. The eFPGA uses 36 GPIOs to interact with the accelerator using a custom interface.

In the first phase, the eFPGA sends the weights of the RNN-model into the four chips. Then, for every sample acquired by the MCU I/O subsystem, the CPU extracts the Mel-Frequency Cepstral Coefficients (MFCCs). In parallel, the eFPGA autonomously fetches the coefficients from the main memory of the MCU and sends them to the off-chip accelerator. Once the inference on the accelerator has been computed, the result is sent back to the eFPGA, which stores it to the main memory of the MCU and finally notifies the core with an interrupt. Fig. 5 shows the data flow from the microphone to the accelerator and back to the MCU. The utilization of the eFPGA is only 10%. Managing 36 GPIOs through MCU firmware (of which one is actually the clock of the off-chip accelerator) would require the core to run at higher frequency than the eFPGA due to the sequential nature of software. In this example the external accelerator is running at 80 MHz. This means that in the best case, the

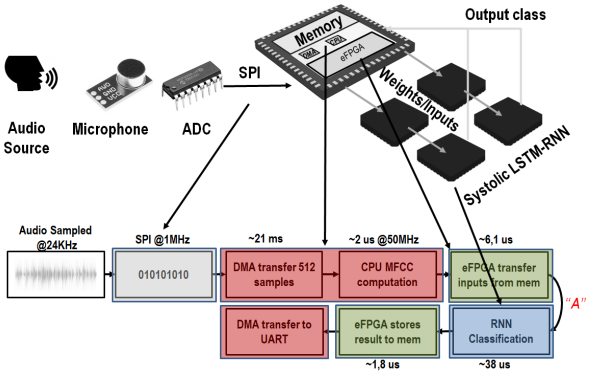


Figure 5. Example of an application where the proposed design is driving custom protocol off-chip accelerators. Data coming from microphones are first pre-processed by the MCU, then sent to the off-chip accelerator via eFPGA for classification.

CPU should be able to perform ~7 operations in 12.5 ns, which requires 560 MHz, and 2.5× higher energy consumption than the eFPGA based solution.

6.3 CPU subsystem accelerator

In the context of on-the-edge computation, accelerators are used to increase performance and the energy efficiency of

Table 3
Performance comparison with state-of-the-art MCU and eFPGA systems.

	Borgatti [28]	**Lodi** [29]	**Renzini** [30]	**Fournaris** [63]	**Whatmough** [49]	**Bol** [12]	**This Work**
Technology [nm]	180	130	90	65	16	28	22
I$/D$/SRAM [kB]	8/8/48	8/8/256	-/-/32	8/-/656	2K[1]/-/4K	-/-/64	-/-/512
Voltage Range [V]	1.8	1.2	1.2	1.2	0.5 - 1.0	0.4 - 0.8	0.5 - 0.8
FPGA IP macro	Hard	Hard	Soft	Hard	Hard	-	Hard
FPGA Area [mm^2]	8.2	6.0	0.347	-	1.0	-	4.0
FPGA #LUT	15 kGE	15 kGE	96 5/4:2	12084 4:1[2]	8800 6:2[3]	-	6018 4:1
FPGA #FF	-	-	192	12084[2]	22656[4]	-	4096
FPGA #DSP	-	-	-	22[5]	80 MACs[6]	-	2 vecMACs
Access Mode to SoC	GPIOs m/s mmap RX DMA	GPIOs s mmap TX/RX DMA	s mmap	GPIOs m/s mmap TX/RX DMA	m/s mmap	- - -	GPIOs m/s mmap TX/RX DMA
FPGA Lkg Power *	-	-	-	-	12000[7]	-	20.5 - 2178
FPGA Max Freq.**	175	166	50	160	734	-	475
FPGA Power Density ***	-	-	34.72@1.2V[8]	962@1.2V[9]	-	-	31.98@0.6V[10]
MCU Lkg Power *	-	-	-	7000[11]	-	1 - 30	532 - 2386
MCU Max Freq. **	175	166	50	166	-	80	600
MCU Power Density ***	-	-	101.22@1.2V[8]	31@1.2V[12]	-	3@0.4V, 48MHz	11.88@0.49V, 135MHz
MCU+FPGA Power Density ***	-	1807.23@1.8V	135.94@1.2V[8]	993@1.2V[9,12]	-	-	46.83@0.52V[13]

* Power numbers are in µW ** Frequency numbers are in MHz *** Power density numbers are in µW/MHz
[1] Two 64 kB of L1 cache shared between Instructions and Data for each core, plus 2 MBytes of L2 cache.
[2] SmartFusion2 M2S010S data available in the product brief.
[3] 2520×2 LUTs for the two logic tile and 1088×2 for the two DSP tiles [71].
[4] 6304×2 flip-flops for two logic tile. 5024×2 for the two DSP tiles [71].
[5] Signed multiplication, dot product, and built-in addition, subtraction, and accumulation units.
[6] 40×2 MACs for the two DSP tile [71].
[7] 3 mW reported in the datasheet [71]. Assuming it is for a 1×1 tile, [49] uses a 2×2 tile, thus 12 mW have been reported in the Table.
[8] Average measurements.
[9] Estimated from [63]. It assumes the FPGA runs at 160 MHz. [10] When *FF2SOC* design is synthesized on the eFPGA [11] Includes FPGA leakage power as well.
[12] Number taken from [63]. The authors use the ARM Cortex M3 power consumption from the datasheet reported in 90 nm LP.
[13] When *FF2SOC* design is synthesized and running on the eFPGA and the MCU is computing a matrix multiplication at the same time

Table 4
Resource utilization, power consumption and overall energy savings for implementing different use-cases on the eFPGA.

Use Case	**GPIO**	**FF**	**LUT**	**Power [mW]**	**Energy Saving [×]**
Custom I/O	36	205	289	6.0	2.5
BNN	0	854	1229	12.5	2.2
CRC	0	20	47	7.5	42.2

such devices [79]. For pattern recognition tasks in the visual domain, deep quantized neural networks are an attractive model due to its limited memory and computational requirements [80]. In extreme cases, single-bit representation for weights and data is chosen to minimize the memory footprint and the computational resources, as it requires simple operations as logic XOR rather than multiplications to compute convolutions. Such neural networks are called Binary Neural Networks (BNN) [81], [82]. The eFPGA has sufficient resources to allow these accelerators to be implemented, freeing the core for other computing tasks.

The BNN accelerator designed for this scope has four interfaces towards the main memory to maximize the bandwidth, and it is a simplified version of the accelerator presented in [13]. It assumes that input layers and filters are organized as a 3D array (number of filters × rows × columns) of integers, where each integer represents a 32 one-bit channels. The accelerator is implemented to operate on two 3×3 windows with eight filters f0, ..., f7 in parallel to simplify the controlling part, but this is not a limiting factor for the use-case under study. The accelerator is programmed via the APB interface by the core with the output, input and filter layer pointers, the number of rows and columns of the input layer, and with the START command. The eFPGA starts by fetching two 32 bit input elements, then four 32 bit elements are fetched in parallel twice to acquire the eight filter elements.

The eFPGA performs the XOR function between the inputs and the eight filters, accumulates all the single-bit partial results. The sixteen 3×3 convolution results are then compared with a programmed threshold to compute the activation functions. The accelerator autonomously iterates over the input rows and columns; then, it sends an interrupt to the core to signal the end of the computation. During this

period, the core can wait for the accelerator to finish in IDLE mode to save power or deal with other tasks in parallel (for example scheduling the next I/O tasks, elaborating previously filtered data, etc.). The design occupies 42% of the SLCs available, and it uses 4 memory interfaces, the APB port, and it generates 1 event. The application consumes 12.5 mW (eFPGA+MCU), and it runs in 371 µs at 125 MHz. Although the core implements custom instructions to speed up such kernels (as the pop count instruction), and it can run faster (600 MHz against 125 MHz), to implement the same function the CPU consumes 15 mW, and it runs in 675 µs, with an energy efficiency 2.2× lower than the eFPGA.

As a second CPU accelerator, a cyclic redundancy check (CRC) accelerator has been implemented in the eFPGA to ensure data integrity and error correction [83]. Such an accelerator uses the I/O DMA interface to leverage the linear address generator already present in the μDMA and thus saving resources in the eFPGA. The CPU programs the μDMA to fetch data from the L2 memory and transmits them to the eFPGA accelerator, which calculates the CRC value. The accelerator has a register to know the number of data to process, whereas the read- and write-pointers are written in the μDMA configuration registers. This low area accelerator consumes only 2% of the SLCs available, and it only uses 1 interface towards the µDMA with configuration, TX/RX ports. The application consumes only 7.5 mW (eFPGA+MCU), and it runs in 3.7 µs at 193 MHz for 1024 byte data. The CPU consumes 15 mW, and it runs in 78 µs, with an energy efficiency 42.2× less than the eFPGA. To compare the performance of the proposed eFPGA-based system with respect to the Microsemi PolarFire IoT gateway-class FPGA SoC [56], the power estimator from Microsemi has been used. Results show a power consumption of 111 mW, 14.8× higher than our work. The estimation has been performed setting the same frequency, number of LUTs and flip-flops.

Table 4 shows the number of GPIOs, number of flip-flops (FF), and LUTs required by each use case. Power figures (expressed in mW) correspond to the system when the eFPGA runs, and the CPU waits for the result, whereas the final column shows the energy gained by running the accelerator on the eFPGA rather than software. In the Custom I/O example, the SW could not handle the protocol at the speed required, for that example eFPGA was the only viable solution.

Basic interfaces like I2C and UART have been implemented on the eFPGA using the DMA interface with about 5% of eFPGA resources, and a more complex parallel camera interface with full DMA support implementation uses only 12% of available eFPGA resources.

Comparison with SoA

Table 3 shows a comparison with various chips reported in the literature. The table includes heterogeneous reconfigurable systems composed of MCU and eFPGA, an embedded domain FPGA SoC, and an advanced low-power MCUs in 28 nm FDSOI. The standalone MCU [12] has a 4× smaller power density (µW/MHz). However, our MCU features 8x larger memory capacity and significantly larger peak performance as well: 7.5× higher maximum frequency, 3.19 vs. 2.33 Coremark/MHz, and almost 6× better performance in near-sensor processing workloads when compared to the ARM Cortex M0 processor used in [12]. Hence, our energy efficiency on the targeted application domain is 1.5× better.

The advanced MCU+eFPGA system presented in [49] is a high-performance class system implemented in 25 mm^2, where a bigger eFPGA (6× higher leakage power), two application class 64 bit cores, a quad-core cluster accelerator, and 12× bigger memory are used (including caches). The eFPGA offers 80 MACs blocks, more LUTs, and eFPGA flip-flops, and provides remarkable energy efficiency of 312 GOPS/W. Thanks to the abundance of DSP blocks in the FPGA fabric. However, this system is meant to be used in high-performance applications consuming higher dynamic and leakage power not suitable for IoT applications. On the other hand, Arnold, although achieving a lower peak efficiency, is in a power range suitable for IoT applications (below hundreds of mW). Moreover, the reverse body biasing applied to the FPGA fabric can reduce leakage power to a value as low as 20.5 µW, more than two orders of magnitude better than [49]. The Microsemi SmartFusion2 SoC [41] used in [63] is built in 65 nm. The whole system can run up to 160 MHz (> 3.75× slower than the proposed work), and it achieves 21× higher power density. The works of Borgatti [28] and Lodi [29] exploit embedded reconfigurable datapaths to accelerate DSP patterns of signal processing applications, achieving remarkable performance and operating frequency despite the old nodes used for implementation. With respect to these works and the other heterogeneous MCU+eFPGA systems of the same class [28], [29], [30], the proposed SoC has more than 2.9× better efficiency, more than 3.4× better performance, and more than 2.2× larger capacity. Moreover, this is the first design offering flexible connections enabling reconfigurable peripherals, I/O accelerators, shared-memory accelerators, and supporting state-retentive deep sleep based on reverse body bias, paving the way for flexible fully programmable IoT end-nodes.

7 Conclusion

In this paper, we presented *Arnold*; a RISC-V based MCU extended with an embedded FPGA for flexible power-constraints energy-efficient IoT devices. The system has built-in GF22FDX, it occupies 9 mm^2, and it leverages body bias to tune performance-power trades off. The eFPGA is a 32×32 array macro provided by QuickLogic connected to the rest of the system through four parallel memory interfaces (128 bit per transaction); a TX/RX I/O DMA interface; sixteen events to interact with the CPU; GPIOs; and APB. The paper shows how the eFPGA can be used to extend and accelerate the SoC peripheral subsystem, as well as a CPU accelerator. The eFPGA has more than 6K LUTs and 4K flip-flops, enough to implement standard and custom peripherals used in the IoT domain and simple accelerators to enhance the energy efficiency of the SoC. It achieves 46.83 µW/MHz, top in class in the mW domain of IoT devices. The CPU runs up to 600 MHz (620 with FBB), more than 7× faster than the best energy efficient MCU. Leakage power of the whole system can be as low as 552 µW when the MCU runs at 0.5 V, and the eFPGA is kept in state retentive deep-sleep via RBB. The paper shows that integrating an eFPGA in an MCU in GF22FDX gives to IoT devices

the high versatility needed for extended product life and shorter time-to-market, still without waiving performance, power and energy efficiency.

ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 732631, project "OPRECOMP".

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012, pp. 13–16.

[2] A. A. Shkel and E. S. Kim, "Continuous health monitoring with resonant-microphone-array-based wearable stethoscope," IEEE Sensors Journal, vol. 19, no. 12, pp. 4629–4638, June 2019.

[3] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8357–8371, Oct 2019.

[4] W. Xia, Y. Zhou, X. Yang, K. He, and H. Liu, "Toward portable hybrid surface electromyography/a-mode ultrasound sensing for human–machine interface," IEEE Sensors Journal, vol. 19, no. 13, pp. 5219–5228, July 2019.

[5] A. J. Casson, D. C. Yates, S. J. M. Smith, J. S. Duncan, and E. Rodriguez-Villegas, "Wearable electroencephalography," IEEE Engineering in Medicine and Biology Magazine, vol. 29, no. 3, pp. 44–56, May 2010.

[6] D. Rossi, C. Mucci, M. Pizzotti, L. Perugini, R. Canegallo, and R. Guerrieri, "Multicore signal processing platform with heterogeneous configurable hardware accelerators," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 9, pp. 1990–2003, 2013.

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

[8] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gürkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beignè, F. Clermidy, P. Flatresse, and L. Benini, "Energy-efficient near-threshold parallel computing: The pulpv2 cluster," IEEE Micro, vol. 37, no. 5, pp. 20–31, Sep. 2017.

[9] A. Pullini, D. Rossi, I. Loi, G. Tagliavini, and L. Benini, "Mr.wolf: An energy-precision scalable parallel ultra low power soc for iot edge processing," IEEE Journal of Solid-State Circuits, vol. 54, no. 7, pp. 1970–1981, July 2019.

[10] D. Bol, J. De Vos, C. Hocquet, F. Botman, F. Durvaux, S. Boyd, D. Flandre, and J. Legat, "Sleepwalker: A 25-mhz 0.4-v sub-mm² 7- μW/MHz microcontroller in 65-nm lp/gp cmos for low-carbon wireless sensor nodes," IEEE Journal of Solid-State Circuits, vol. 48, no. 1, pp. 20–32, Jan 2013.

[11] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gürkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beignè et al., "Energy-efficient near-threshold parallel computing: The pulpv2 cluster," Ieee Micro, vol. 37, no. 5, pp. 20–31, 2017.

[12] D. Bol, M. Schramme, L. Moreau, T. Haine, P. Xu, C. Frenkel, R. Dekimpe, F. Stas, and D. Flandre, "19.6 a 40-to-80mhz sub-4µw/mhz ulv cortex-m0 mcu soc in 28nm fdsoi with dual-loop adaptive back-bias generator for 20µs wake-up from deep fully retentive sleep mode," in 2019 IEEE International Solid- State Circuits Conference - (ISSCC), Feb 2019, pp. 322–324.

[13] F. Conti, P. D. Schiavone, and L. Benini, "Xnor neural engine: A hardware accelerator ip for 21.6-fj/op binary neural network inference," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2940–2951, Nov 2018.

[14] R. Bansal and A. Karmakar, "Closely-coupled lifting hardware for efficient dwt computation in an soc," Journal of Signal Processing Systems, pp. 1–13, 2019.

[15] M. Cavalcante, F. Schuiki, F. Zaruba, M. Schaffner, and L. Benini, "Ara: A 1-ghz+ scalable and energy-efficient risc-v vector processor with multiprecision floating-point support in 22-nm fd-soi," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 2, pp. 530–543, Feb 2020.

[16] T. Fritzmann, U. Sharif, D. Müller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum co-processors based on risc-v," in 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2019, pp. 1148–1153.

[17] A. Jafari, A. Ganesan, C. S. K. Thalisetty, V. Sivasubramanian, T. Oates, and T. Mohsenin, "Sensornet: A scalable and low-power deep convolutional neural network for multimodal data classification," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 1, pp. 274–287, 2019.

[18] X. Yu, Y. Wang, J. Miao, E. Wu, H. Zhang, Y. Meng, B. Zhang, B. Min, D. Chen, and J. Gao, "A data-center fpga acceleration platform for convolutional neural networks," in 2019 29th International Conference on Field Programmable Logic and Applications (FPL), Sep. 2019, pp. 151–158.

[19] A. Ibrahim and M. Valle, "Real-time embedded machine learning for tensorial tactile data processing," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 11, pp. 3897–3906, 2018.

[20] H. Chen, S. Madaminov, M. Ferdman, and P. Milder, "Sorting large data sets with fpga-accelerated samplesort," in 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2019, pp. 326–326.

[21] A. G. Sawant, V. N. Nitnaware, and A. A. Deshpande, "Spartan-6 fpga implementation of aes algorithm," in ICCCE 2019. Springer, 2020, pp. 205–211.

[22] J. Liao, M. Jost, M. Schaffner, M. Magno, M. Korb, L. Benini, F. Tebbenjohanns, R. Reimann, V. Jain, M. Gross, A. Militaru, M. Frimmer, and L. Novotny, "Fpga implementation of a kalman-based motion estimator for levitated nanoparticles," IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 7, pp. 2374–2386, July 2019.

[23] H. Homulle, S. Visser, and E. Charbon, "A cryogenic 1 gsa/s, soft-core fpga adc for quantum computing applications," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 11, pp. 1854–1865, 2016.

[24] A. Jafari, N. Buswell, M. Ghovanloo, and T. Mohsenin, "A low-power wearable stand-alone tongue drive system for people with severe disabilities," IEEE Transactions on Biomedical Circuits and Systems, vol. 12, no. 1, pp. 58–67, Feb 2018.

[25] P. Anagnostou, A. Gomez, P. Hager, H. Fatemi, J. P. de Gyvez, L. Thiele, and L. Benini, "Torpor: A power-aware hw scheduler for energy harvesting iot socs," in 2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), July 2018, pp. 54–61.

[26] V. Rosello, J. Portilla, and T. Riesgo, "Ultra low power fpga-based architecture for wake-up radio in wireless sensor networks," in IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society, Nov 2011, pp. 3826–3831.

[27] I. Williams, S. Luan, A. Jackson, and T. G. Constandinou, "Live demonstration: A scalable 32-channel neural recording and real-time fpga based spike sorting system," in 2015 IEEE Biomedical Circuits and Systems Conference (BioCAS), Oct 2015, pp. 1–5.

[28] M. Borgatti, F. Lertora, B. Foret, and L. Cali, "A reconfigurable system featuring dynamically extensible embedded microprocessor, fpga, and customizable i/o," IEEE Journal of Solid-State Circuits, vol. 38, no. 3, pp. 521–529, March 2003.

[29] A. Lodi, A. Cappelli, M. Bocchi, C. Mucci, M. Innocenti, C. De Bartolomeis, L. Ciccarelli, R. Giansante, A. Deledda, F. Campi, M. Toma, and R. Guerrieri, "Xisystem: a xirisc-based soc with reconfigurable io module," IEEE Journal of Solid-State Circuits, vol. 41, no. 1, pp. 85–96, Jan 2006.

[30] F. Renzini, C. Mucci, D. Rossi, E. F. Scarselli, and R. Canegallo, "A fully programmable efpga-augmented soc for smart power applications," IEEE Transactions on Circuits and Systems I: Regular Papers, pp. 1–13, 2019.

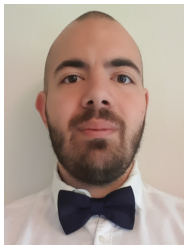
[31] NXP: Power consumption and measurement of i.MXRT1050. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN12094.pdf

[32] STMicroelectronics: STM32L476xx datasheet. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32l476je.pdf

[33] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, "Quentin: an ultra-low-power pulpissimo soc in

22nm fdx," in 2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S). IEEE, 2018, pp. 1–3.
[34] Microsemi IGLOO nano Low Power Flash FPGAs datasheet. [Online]. Available: Availableonlineathttps://www.microsemi.com
[35] Lattice Semiconductor iCE40 UltraLite Family Data Sheet. [Online]. Available: Availableonlineathttp://www.latticesemi.com
[36] Menta: eFPGA IP Cores. [Online]. Available: Availableonlineathttps://www.menta-efpga.com/efpga-ips
[37] Achronix Speedcore eFPGA. [Online]. Available: SpeedcoreFPGAdatasheetavailableonline
[38] Flex-Logix eFPGA. [Online]. Available: https://flex-logix.com/efpga/
[39] Quicklogic: ArcticPro 2 eFPGA. [Online]. Available: Availableonlineathttps://www.quicklogic.com/products/efpga/arcticpro-2/
[40] NXP: i.MX 7ULP Applications Processor Consumer Products. [Online]. Available: Datasheetavailableonline
[41] SmartFusion: SmartFusion2 SoC. [Online]. Available: https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2
[42] Silicon Labs: EFM32 Giant Gecko 11 32bit Microcontrollers. [Online]. Available: Datasheetavailableonline
[43] Xilinx: Virtex Ultrascale+. [Online]. Available: Availableonlineathttps://www.xilinx.com
[44] Intel Cyclone 10 GX Device Overview. [Online]. Available: Availableonlineathttps://www.intel.com
[45] GreenWaves Technology: GAP8 PRODUCT BRIEF. [Online]. Available: GAP8productbriefavailableonline
[46] F. Conti, R. Schilling, P. D. Schiavone, A. Pullini, D. Rossi, F. K. Gürkaynak, M. Muehlberghuber, M. Gautschi, I. Loi, G. Haugou, S. Mangard, and L. Benini, "An iot endpoint system-on-chip for secure and energy-efficient near-sensor analytics," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 9, pp. 2481–2494, Sep. 2017.
[47] Xilinx: Zynq-7000 SoC. [Online]. Available: Xilinxds190-Zynq-7000datasheetavailableonline
[48] Intel: Arria V SOC FPGAS. [Online]. Available: IntelArriaVSOCFPGAdatasheetavailableonline
[49] P. N. Whatmough, S. K. Lee, M. Donato, H. Hsueh, S. Xi, U. Gupta, L. Pentecost, G. G. Ko, D. Brooks, and G. Wei, "A 16nm 25mm2 soc with a 54.5x flexibility-efficiency range from dual-core arm cortex-a53 to efpga and cache-coherent accelerators," in 2019 Symposium on VLSI Circuits, June 2019, pp. C34–C35.
[50] P. Davide Schiavone, F. Conti, D. Rossi, M. Gautschi, A. Pullini, E. Flamand, and L. Benini, "Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications," in 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Sep. 2017, pp. 1–8.
[51] H.-C. Ng, C. Liu, and H. K.-H. So, "A soft processor overlay with tightly-coupled fpga accelerator," arXiv preprint arXiv:1606.06483, 2016.
[52] C. Heinz, Y. Lavan, J. Hofmann, and A. Koch, "A catalog and in-hardware evaluation of open-source drop-in compatible risc-v softcore processors," in IEEE Proc. International Conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE, 2019.
[53] R. Höller, D. Haselberger, D. Ballek, P. Rössler, M. Krapfenbauer, and M. Linauer, "Open-source risc-v processor ip cores for fpgas—overview and evaluation," in 2019 8th Mediterranean Conference on Embedded Computing (MECO). IEEE, 2019, pp. 1–6.
[54] Y. Choi, K. You, J. Choi, and W. Sung, "A real-time fpga-based 20000-word speech recognizer with optimized dram access," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 8, pp. 2119–2131, 2010.
[55] A. Lindoso, L. Entrena, M. García-Valderas, and L. Parra, "A hybrid fault-tolerant leon3 soft core processor implemented in low-end sram fpga," IEEE Transactions on Nuclear Science, vol. 64, no. 1, pp. 374–381, Jan 2017.
[56] PolarFire SoC Advance Product Overview . [Online]. Available: https://www.microsemi.com/product-directory/soc-fpgas/5498-polarfire-soc-fpga#resources
[57] B. Pandey, B. Das, A. Kaur, T. Kumar, A. M. Khan, D. A. Hussain, and G. S. Tomar, "Performance evaluation of fir filter after implementation on different fpga and soc and its utilization in communication and network," Wireless Personal Communications, vol. 95, no. 2, pp. 375–389, 2017.
[58] W. Qiao, Z. Fang, M. F. Chang, and J. Cong, "An fpga-based bwt accelerator for bzip2 data compression," in 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), April 2019, pp. 96–99.
[59] A. Di Mauro, F. Conti, and L. Benini, "An ultra-low power address-event sensor interface for energy-proportional time-to-information extraction," in 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), June 2017, pp. 1–6.
[60] I. Williams, S. Luan, A. Jackson, and T. G. Constandinou, "Live demonstration: A scalable 32-channel neural recording and real-time fpga based spike sorting system," in 2015 IEEE Biomedical Circuits and Systems Conference (BioCAS), Oct 2015, pp. 1–5.
[61] Microsemi: Open. Lowest Power. Programmable RISC-V Solutions. [Online]. Available: https://www.microsemi.com/product-directory/mi-v-embedded-ecosystem/4406-risc-v-cpus
[62] T. Gomes, S. Pinto, T. Gomes, A. Tavares, and J. Cabral, "Towards an fpga-based edge device for the internet of things," in 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA), Sep. 2015, pp. 1–4.
[63] A. P. Fournaris, C. Alexakos, C. Anagnostopoulos, C. Koulamas, and A. Kalogeras, "Introducing hardware-based intelligence and reconfigurability on industrial iot edge nodes," IEEE Design Test, vol. 36, no. 4, pp. 15–23, Aug 2019.
[64] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gürkaynak, and L. Benini, "Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2700–2713, 2017.
[65] A. Waterman, Y. Lee, D. A. Patterson, K. Asanovic, V. I. U. level Isa, A. Waterman, Y. Lee, and D. Patterson, "The risc-v instruction set manual," 2014.
[66] A. Rahimi, I. Loi, M. R. Kakoee, and L. Benini, "A fully-synthesizable single-cycle interconnection network for shared-l1 processor clusters," in 2011 Design, Automation Test in Europe, March 2011, pp. 1–6.
[67] A. Pullini, D. Rossi, G. Haugou, and L. Benini, "μdma: An autonomous i/o subsystem for iot end-nodes," in 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS). IEEE, 2017, pp. 1–8.
[68] F. Zaruba, F. Schuiki, S. Mach, and L. Benini, "The floating point trinity: A multi-modal approach to extreme energy-efficiency and performance," in 26th IEEE International Conference on Electronics Circuits and Systems, 2019.
[69] A. Di Mauro, F. Conti, P. Schiavone, D. Rossi, and L. Benini, "Pushing on-chip memories beyond reliability boundaries in micropower machine learning applications," in 65th International Electron Devices Meeting (IEDM 2019), 2019.
[70] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gurkaynak, A. Bartolini, P. Flatresse, and L. Benini, "A 60 gops/w, -1.8v to 0.9v body bias ulp cluster in 28nm utbb fd-soi technology," Solid-State Electronics, vol. 117, pp. 170 – 184, 2016, pLANAR FULLY-DEPLETED SOI TECHNOLOGY. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038110115003342
[71] EFLX: EFLX 4K Product Brief for TSMC 12FFC+/12FFC/16FFC+/FFC/FF+. [Online]. Available: Availableonlineathttps://flex-logix.com/efpga/
[72] V. J. Kartsch, S. Benatti, P. D. Schiavone, D. Rossi, and L. Benini, "A sensor fusion approach for drowsiness detection in wearable ultra-low-power systems," Information Fusion, vol. 43, pp. 66–76, 2018.
[73] F. H. Elfouly, M. I. Mahmoud, M. I. Dessouky, and S. Deyab, "Comparison between haar and daubechies wavelet transformations on fpga technology," International Journal of Computer, Information, and Systems Science, and Engineering, vol. 2, no. 1, 2008.
[74] A. Burrello, L. Cavigelli, K. Schindler, L. Benini, and A. Rahimi, "Laelaps: An energy-efficient seizure detection algorithm from long-term human ieeg recordings without false alarms," in 2019 Design, Automation Test in Europe Conference Exhibition (DATE), March 2019, pp. 752–757.
[75] Inivation. (2020) Inivation dynamic vision platform. [Online]. Available: https://inivation.com/dvp/
[76] S. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, "Asynchronous binaural spatial audition sensor with $2 \times 64 \times 4$ channel output," IEEE Transactions on Biomedical Circuits and Systems, vol. 8, no. 4, pp. 453–464, Aug 2014.

[77] F. Conti, L. Cavigelli, G. Paulin, I. Susmelj, and L. Benini, "Chipmunk: A systolically scalable 0.9 mm2, 3.08gop/s/mw @ 1.2 mw accelerator for near-sensor recurrent neural network inference," in 2018 IEEE Custom Integrated Circuits Conference (CICC), April 2018, pp. 1–4.
[78] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H. Yoo, "Unpu: An energy-efficient deep neural network accelerator with fully variable weight bit precision," IEEE Journal of Solid-State Circuits, vol. 54, no. 1, pp. 173–185, Jan 2019.
[79] D. Chen, J. Cong, S. Gurumani, W. Hwu, K. Rupnow, and Z. Zhang, "Platform choices and design demands for iot platforms: cost, power, and performance tradeoffs," IET Cyber-Physical Systems: Theory Applications, vol. 1, no. 1, pp. 70–77, 2016.
[80] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," The Journal of Machine Learning Research, vol. 18, no. 1, pp. 6869–6898, 2017.
[81] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in Advances in neural information processing systems, 2015, pp. 3123–3131.
[82] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in European conference on computer vision. Springer, 2016, pp. 525–542.
[83] E. Tsimbalo, X. Fafoutis, and R. J. Piechocki, "Crc error correction in iot applications," IEEE Transactions on Industrial Informatics, vol. 13, no. 1, pp. 361–369, Feb 2017.



Pasquale Davide Schiavone received his B.Sc. (2013) and M.Sc. (2016) in computer engineering from Polytechnic of Turin. In 2016 he has started his Ph.D. studies at the Integrated Systems Laboratory, ETH Zurich. In 2018, he has been Ph.D visiting student in the Centre for Bio-Inspired Technology, Imperial College London. His research interests include datapath blocks design, low-power microprocessors in multi-core systems and deep-learning architectures for energy-efficient systems.

Davide Rossi , received the PhD from the University of Bologna, Italy, in 2012. He has been a post doc researcher in the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi" at the University of Bologna since 2015, where he currently holds an assistant professor position. His research interests focus on energy efficient digital architectures in the domain of heterogeneous and reconfigurable multi and many-core systems on a chip. This includes architectures, design implementation strategies, and runtime support to address performance, energy efficiency, and reliability issues of both high end embedded platforms and ultra-low-power computing platforms targeting the IoT domain. In these fields he has published more than 100 papers in international peer-reviewed conferences and journals. He is recipient of Donald O. Pederson Best Paper Award 2018.

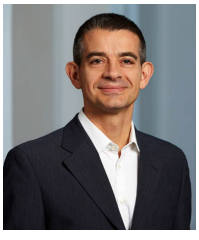
Alfio Di Mauro received his M.Sc. degree in Electronic Engineering from the Electronics and Telecommunications Department (DET) of Politecnico di Torino in 2016. In January 2017, he started to work as researcher assistant in the Integrated System Laboratory (IIS) of the Swiss Federal Institute of Technology of Zurich, in the group led by Prof. Luca Benini. Since September 2017, he is pursuing the PhD in electrical engineering in the same Laboratory. His research is mainly focused on the design of digital Ultra-Low Power (ULP) System-on-Chip (SoC) for Event-Driven edge computing.

Frank K. Gürkaynak has obtained his B.Sc. and M.Sc, in electrical engineering from the Istanbul Technical University, and his Ph.D. in electrical engineering from ETH Zürich in 2006. He is currently working as a senior researcher at the Integrated Systems Laboratory of ETH Zürich. His research interests include digital low-power design and cryptographic hardware.

Timothy Saxe (Ph.D.) joined QuickLogic in May 2001. Dr. Saxe has served as our Senior Vice President of Engineering and Chief Technology Officer since August 2016 and Senior Vice President and Chief Technology Officer since November 2008. Previously, Dr. Saxe has held a variety of executive leadership positions in QuickLogic including Vice President of Engineering and Vice President of Software Engineering. Dr. Saxe was Vice President of Flash Engineering at Actel Corporation, a semiconductor manufacturing company, from November 2000 to February 2001. Dr. Saxe joined GateField Corporation, a design verification tools and services company formerly known as Zycad, in June 1983 and was a founder of their semiconductor manufacturing division in 1993. Dr. Saxe became GateField's Chief Executive Officer in February 1999 and served in that capacity until Actel Corporation acquired GateField in November 2000. Dr. Saxe holds a B.S.E.E. degree from North Carolina State University, and an M.S.E.E. degree and a Ph.D. in Electrical Engineering from Stanford University..

Mao Wang is a Sr. Director of Product at QuickLogic, with the mission of democratizing embedded FPGA into every SoC. He holds a B.S. degree en Electrical Engineering and a M.S. degree in Engineering Management from Santa Clara University.

Ket Chong Yap joined QuickLogic in September 1999, actively participating in QuickLogic FPGA product development. Prior to joining QuickLogic, Mr. Yap was with EXEL Microelectronics as Quality Assurance Engineer from 1990 to 1991, Product/Test Engineer from 1992 to 1994, and Design Engineer from 1995 to 1996, working on EEPROM technology. Mr. Yap was also involved with Programmable Microelectronics Corporation from 1997 to 1998, working on FLASH memory. Mr. Yap hold a B.S. degree in Electrical Engineering from the Iowa State University, Ames.



Luca Benini (F'07) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1997. He has served as the Chief Architect of the Platform 2012/STHORM Project with STMicroelectronics, Grenoble, France, from 2009 to 2013. He held visiting/consulting positions with École Polytechnique Fédérale de Lausanne, Stanford University, and IMEC. He is currently a Full Professor with the University of Bologna, Bologna, Italy. He has authored over 700 papers in peer-reviewed international journals and conferences, four books, and several book chapters. His current research interests include energy-efficient system design and multicore system-on-chip design. Dr. Benini is a member of Academia Europaea. He is currently the Chair of Digital Circuits and Systems with ETH Zürich, Zürich, Switzerland.