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 K. Gürkaynak,
Timothy Saxe, Mao Wang, Ket Chong Yap, and 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-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 (eFPGA). We demonstrate the flexibility of the system-on-chip (SoC) to tackle the challenges of many emerging IoT applications, such as interfacing sensors and accelerators with nonstandard interfaces, performing on-the-fly preprocessing tasks on data streamed from peripherals, and 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 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 reconfigurable SoCs of the same class.**

***Index Terms*—Edge computing, embedded systems, field-programmable gate array (FPGA), Internet of Things (IoT), microcontroller, open source, RISC-V.**

I. Introduction

THE end nodes of the Internet of Things (IoT) require energy-efficient, powerful, and flexible ultralow-power computing platforms to deal with a wide range of near-sensor applications [1]. These system-on-chips (SoCs) must be able to connect to low-power sensors such as arrays of microphones [2], cameras [3], and 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, extract only relevant information (e.g., [5]), or filter, encrypt, and anonymize data. Analyzing and distilling information as it moves 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 field-programmable gate arrays (FPGAs), or as software programmable on microcontroller units (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 due to their high versatility, low power, and low cost. State-of-the-art (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 [8]–[11], 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 the 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 [12], frequency-domain transforms [13], linear algebra [14], and security engines [15]. 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 more specialized design skills), 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 [16]–[18], sorting [19], and cryptography accelerators for data centers

Manuscript received August 26, 2020; revised December 28, 2020; accepted January 31, 2021. Date of publication March 4, 2021; date of current version April 1, 2021. This work was supported by the European Union's Horizon 2020 Research and Innovation Program through project "OPRECOMP" under Grant 732631. *(Corresponding author: Pasquale Davide Schiavone.)*

Pasquale Davide Schiavone, Alfio Di Mauro, Frank K. Gürkaynak, and Luca Benini are with the Integrated Systems Laboratory, Department of Information Technology and Electrical Engineering (D-ITET), ETH Zürich, 8092 Zürich, Switzerland (e-mail: pschiavo@iis.ee.ethz.ch).

Davide Rossi is with the Energy-Efficient Embedded Systems Laboratory, Department of Electrical, Electronic and Information Engineering (DEI), University of Bologna, 40126 Bologna, Italy.

Timothy Saxe, Mao Wang, and Ket Chong Yap are with QuickLogic Corporation, San Jose, CA 95131 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3058162.

Digital Object Identifier 10.1109/TVLSI.2021.3058162

[20] to smart instruments [21], analog-to-digital converters [22], low-power systems for wearable applications [23], and control-logic systems [24], and for implementing smart peripherals connected to SoCs [25], [26].

While FPGAs have been traditionally designed as standalone components, a new class of devices featuring FPGAs extended with central processing units (CPUs), epitomized by the highly successful Zynq "all programmable SoC" product family [27], penetrated the embedded devices market during the last decade. These heterogeneous SoCs make the split of computational and control tasks between hardware and software more comfortable, leveraging parallel computing of the FPGA resources in compound with the support of operating systems. However, such SoA SoCs are meant for high-end embedded applications, being much more power-hungry (between few hundreds of mW and watts) than always-ON IoT end nodes (few tens of mW). Recently, the increased integration density of modern silicon technologies allowed a reasonably sized FPGA array (so-called embedded FPGAs or eFPGAs) to be integrated as part of mid-end SoCs such as the Microsemi SmartFusion2, still exceeding power constraints of most IoT end-nodes applications [28]. Moreover, existing solutions have limited options for integrating the eFPGAs at the system level, designed for standalone operation, connected to I/O peripherals such as PCIe to loosely coupled memories as DDR, and either low-bandwidth or high latency system bus such as the AMBA AHB or AXI, respectively.

In this article, we present *Arnold*: a RISC-V-based MCU extended with an eFPGA for always-ON edge-computing systems, implemented in Globalfoundries GF22FDX (GF22FDX) technology, tackling the challenges highlighted above. The SoC targets the mW-range "IoT end-node" MCU profile, where high processing capabilities need to be coupled with low-idle power consumption, high energy efficiency, and high versatility. To this end, in addition to more traditional connections to the I/O PADs, we propose a novel approach to integrate the eFPGA through a high-bandwidth, low-latency interconnect, enabling data sharing with the cores in a single cycle, and with an autonomous I/O DMA subsystem, enabling heterogeneous computing both on tightly coupled memory and I/O data. Moreover, to address the power–performance scalability challenge, power domains and technology knobs for leakage power and performance, such as body biasing, are exposed at the system level to switch-OFF or reduce the eFPGA leakage power consumption (its major contribution) while keeping the state during idle periods by up to 18× and to dramatically scale the energy/performance of the processing system to deal with computing peak requirements. Fig. 1 shows a high-level view of the Arnold architecture with the eFPGA connections highlighted.

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

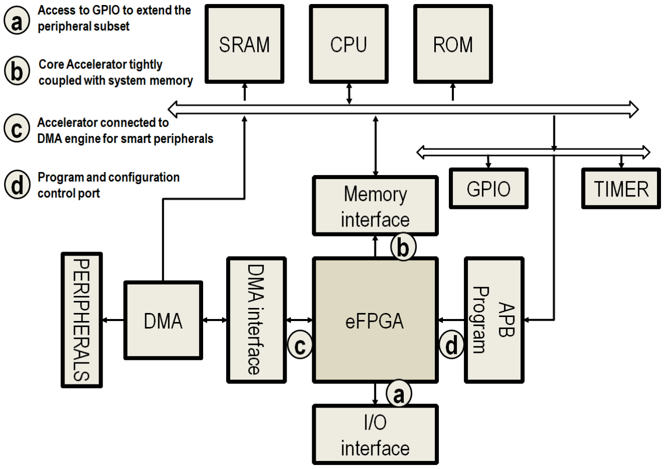

a Access to GPIO to extend the peripheral subset
b Core Accelerator tightly coupled with system memory
c Accelerator connected to DMA engine for smart peripherals
d Program and configuration control port
SRAM
CPU
ROM
GPIO
TIMER
Memory interface
b
PERIPHERALS
DMA
DMA interface
c
eFPGA
d
APB Program
a
I/O interface

Fig. 1. MCU-eFPGA SoC architecture. eFPGA connections toward the MCU and to the external peripherals are highlighted.

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 dataplane: 1) 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; 2) 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; and 3) a direct GPIO interface to implement master or slave peripheral ports for nonstandard off-chip digital sensors or actuators. On the control plane, we provide: 1) an AMBA Advanced Peripheral Bus (APB) interface to allow the user to configure the mapped soft-hardware and 16 interrupts to notify the CPU.
2) *Power Management:* Due to reverse body-biasing (RBB) enabled by conventional-well FDSOI technology used for the physical implementation of the eFPGA fabric, leakage power can be reduced by 18× to 20.5 μW (featuring a fully state retentive bitstream) when eFPGA functionality is not required.
3) *Power and Energy Efficiency:* For IoT end-node mission profiles, the SoC achieves SoA performance and efficiency, leveraging a voltage and frequency scalable architecture from 0.5 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, within a power envelope of 22 mW. 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 [29]–[31].
4) *Use Cases:* We demonstrate the high performance and flexibility of the proposed system on a set of use cases where we exploit the eFPGA subsystem as an I/O subsystem accelerator, a custom I/O peripheral, and a tightly coupled CPU accelerator, improving the energy efficiency of the system from 2.2× to 42.2×.

The remainder of this article 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 Sections IV and V, the software and tools for the proposed SoC, its physical design, and silicon

TABLE I

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	[11], [32], [33], [34]	**Low Power**	[35], [36]	**StandAlone**	[37], [38], [39], [40]
SW Accelerator	[9], [41]	**Low Power SoC**	[28]	**MCU SoC**	[29], [30], [31],
HW Accelerator	[12], [42]	**HP**	[43]		**This Work**
HW/SW Accelerator	[44], [45]	**HP SoC**	[27]	**HP SoC**	[46]

measurements are described respectively, whereas, in Section VI, use cases for the proposed work are reported as application examples. This article concludes in Section VII.

II. Related Work

In this section, we review devices that define the boundaries of its design space: MCUs, FPGAs, eFPGAs, and heterogeneous reconfigurable SoCs. Table 1 shows a summary of related works.

A. Microcontroller Units

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 CPUs based on ARM Cortex-M family of cores, such as the NXP i.MXRT1050 [32], the STMicroelectronics STM32L476xx family [33], 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 ultralow-power (3 μW/MHz) [11] and performance (938 MHz) [34] 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 [12] 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 [41], 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 eight-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 [47]. Finally, heterogeneous systems, such as GAP-8 from GreenWaves Technologies [44] and Fulmine [45], 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 nonstandard sensors.

B. Field-Programmable Gate Arrays

FPGAs are reconfigurable devices that can exploit spatial computations typical of ASIC designs but still retain programmability. They range from high-end FPGAs used for acceleration of high-performance workloads to ultralow-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 Xilinx Zynq-7000 SoC [27], have millions of lookup tables (LUTs), flip-flops (FFs), DSP-blocks, CPU, and SRAM macros containing megabytes of memory. They have typical power consumption in the order of tens of watts [48], and they are usually used as high-performance accelerators on servers connected via Ethernet or PCI interfaces [49].

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 [35], which has up to 3k logic elements,[1] or the Lattice Semiconductor iCE40 UltraLite [36], 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 [50], even with simple data preprocessing engines to lower the bandwidth coming from sensors [23], [26]. In the low-end space, FPGAs can also be extended with hard or soft CPUs to leverage HW/SW codesigned IoT nodes. Hard-CPUs are used in the Microsemi SmartFusion2 SoC 65 nm [28], which proposes an MCU-class (ARM Cortex-M) core running at 166 MHz and an FPGA with DSP blocks and up to 150k logic elements, 656 kB[2] of memory, and power consumption in the order of hundreds of milliwatts. Examples that use the Microsemi SmartFusion2 SoC can be found in [51], which proposes a system where most of the tasks are executed by the ARM core,

[1] One logic element is composed of one four-input LUT and one FF.

[2] 512 bytes of nonvolatile memory.

whereas the FPGA is used for accelerating critical network kernels. In [52], the operating system and user interfaces run in software, whereas the FPGA is used to collect sensor data, extract features, and calculate the nearest neighbor on the extracted information. The system that 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 the 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 Section II-C.

C. eFPGAs

eFPGAs are FPGA IP cores specifically meant to be integrated into SoCs to extend them with programmable logic. Unlike the FPGAs presented in Section II-B, 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, FFs, I/O pins, and so on can be configured. eFPGAs can be provided as soft-IP [31], [37], described in RTL and synthesized with the rest of the system, or hard-IP [29], [30], [46] as hard-macros with predetermined physical layout, featuring a different tradeoff between performance and cost.

For example, in [31], a soft-IP is complementing an MCU for power control applications is implemented using a 90-nm bipolar CMOS DMOS (BCD) technology. This eFPGA is relatively small (only 96 four-input LUTs and 192 FFs) and connected exclusively to the I/O subsystem. Several companies are providing hard-IP blocks, as Achronix [38], which provides 7-nm FinFET eFPGAs, Flex-Logix [39], which provides from 12- to 180-nm eFPGAs macros, QuickLogic Corporation [40], which provides from 22- to 65-nm core IPs, and Menta [37], which provides IPs from 10 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.* [46] presented a 25-mm^2 SoC implemented in 16-nm FinFET technology featuring two ARM A53 cores, a quad-core datapath accelerator, 4-MB on-chip SRAM, and a 2 × 2 FlexLogic eFPGA macro featuring hardwired DSP slices.

In the embedded domain, several solutions have been proposed in different technology nodes. Borgatti *et al.* [29] implemented a 180-nm 20-mm^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 with camera peripherals with preprocessing. Lodi *et al.* [30] 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.

In this work, we propose an SoC featuring an advanced microcontroller augmented by an eFPGA for IoT applications in 22-nm process technology. From an architectural standpoint, the main differentiating feature of the proposed SoC is in how the FPGA has been integrated into the system, being able to act as a tightly coupled memory accelerator for the core via a fast, low-latency, high-bandwidth (128 bit) interconnect, and as a I/O preprocessing engine, being connected as a configurable peripheral for the I/O DMA subsystem. The proposed solution provides 3.4× better performance and 2.9× better efficiency than state-of-the-art heterogeneous reconfigurable SoCs, leveraging the wide voltage range supported by the 22-nm GF22FDX technology. One key feature of the SoC with respect to SoA-related works is the unique capability to exploit RBB enabled by the FD-SOI technology to implement a 20.5-μW state-retentive deep-sleep mode for the eFPGA, reducing the power overhead of the eFPGA integration. This point is further discussed in Section V.

III. Arnold Architecture

The proposed system is built around an in-order RISC-V core[3] based on [53], optimized for signal processing, featuring a four-stage pipeline, and achieving 3.19 Coremark/MHz and up to 2.4 8-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) [54]. In addition, the core has been extended with custom instructions to speedup 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 two half-words at a time. With respect to a closed-source ISA such as ARM, the open-source RISC-V ISA allows for custom extensions to achieve higher performance in the targeted applications domain. For example, in [53], the proposed ISA extensions show 10× better performance than the plain RISC-V or OpenRISC ISA. Furthermore, the core presents a higher Coremax/MHz score with respect to the ARM Cortex-M4 when the GNU GCC compiler is used for both the cores (3.19 versus 2.55, respectively).

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 two regions *NAPOT* (including 4-bytes alignment *NA4*) and the top boundary of an arbitrary range *TOR*. The PMP occupies

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

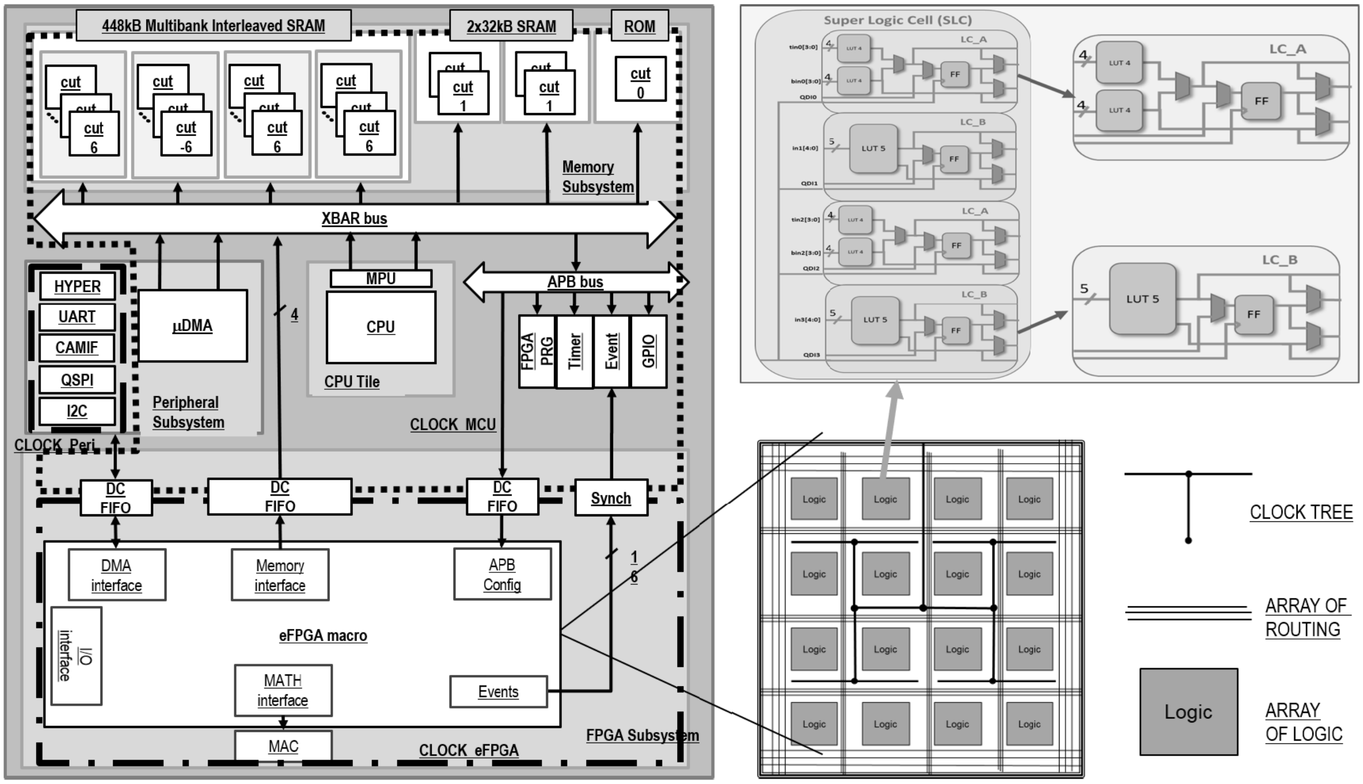


Fig. 2. Detailed block diagram of the proposed design. The eFPGA (bottom) connected with the MCU and its private MAC units in a clock domain (CLOCK eFPGA). Peripherals (center–left) are directly connected to the μ DMA in the peripheral subsystem and operate on the CLOCK Peri clock domain. The rest of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

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.

A. 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 [55] (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 noninterleaved address scheme. Every memory bank is a composition of single-port 4096 × 32 bit words (16 kB) memory cuts optimized for density and power. The chosen interleaving scheme for the four 112-kB (448-kB) memory portion approximates a multiport 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 digital signal processor (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 Gb/s each. In the proposed MCU, they 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 multiport 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.

B. 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 μ DMA based on [56]. The μ DMA is a smart engine that allows peripherals to control transfers to/from memory without the need for the CPU continuous control. It has two ports toward the main memory: one to transmit and one to receive data from peripherals. At 600 MHz, the μ DMA has an aggregated bandwidth equal to 38.4 Mb/s. Except for the JTAG, which is directly connected to a master port of the system bus, the other peripherals are controlled by the μ DMA core, which handles memory requests in a time-multiplexed fashion. The μ DMA control registers are used to select the active peripheral, the peripheral clock frequency, number of transfers, and so on. 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, which is in the same MCU clock domain as the peripherals are not timing critical, and the clock-tree implementation gets simplified.

C. Clock Subsystem

The system includes three compact and energy-efficient frequency-locked loops (FLLs) based on [57]. They take as input an external 32-kHz reference clock and provide internal clocks up to 2.1 GHz. Since *Arnold* does not have any external communication channel that it needs to synchronize to, there are no advantages of using a more complex/costly phase-locked loop (PLL)-based clocking solution. *Arnold* uses one FLL to provide the clock to the eFPGA: one for the peripheral subsystem and one for the remaining modules as CPU, memories, buses, and so on. 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. The user can choose the preferred clock sources.

D. 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 four-mode SoC interface to have the following:

1) an I/O interface with direct connections toward the pad frame of the system, enabling the implementation of custom off-chip interfaces;
2) a memory interface suitable for shared-memory accelerators implemented on the FPGA logic and tightly coupled with the CPU;
3) an I/O DMA interface suitable for implementing I/O filtering functions for data streamed into the system from the standard I/O;
4) 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, and 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, and so on), 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 [55]. 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 four-word dual-clock first-input, first-output (FIFO) 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 [56]. The configuration bus allows controlling the peripherals mapped into the eFPGA with external registers, which can avoid the use of the APB interface described next, and thus save resources. In addition, this interface can be used to stream data through the μ DMA without using eFPGA resources for the address generation logic as it would with the memory interface. In this case, the μ DMA transfers data from the eFPGA to memory (and vice versa) linearly. Communication between the μ DMA and the eFPGA happens using two 32-bit four-word 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 four-word 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 16 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 newly 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 controls the MAC blocks, whether data come 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 reporting 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, and then, the CPU reads the binary array and writes its content to the APB FCB via noncritical load and store instructions.

The eFPGA fabric is organized in four quadrants with dynamic reconfiguration capabilities, each one composed of an array of 16 × 16 superlogic cells (SLCs). Each SLC has four logic cells that are organized in two sublogic 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

TABLE II
Arnold ACTIVE POWER STATES

Power Mode	**Vdd** [V]	**Vdd eFPGA [V]**	**FBB** [V]	**RBB** [V]
eFPGA Off	0.5 - 0.8	0	0 - 0.4	0
eFPGA Retentive	0.5 - 0.8	0.5 - 0.8	0 - 0.4	0 - 1.4
eFPGA On	0.5 - 0.8	0.5 - 0.8	0 - 0.4	0

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 textscoff to save static power.

E. Power Domains

The *Arnold* chip can select different power states to minimize the eFPGA power overhead or maximize performance. The MCU and the eFPGA have different supply voltage pins that are driven by an external power manager. Table II shows the available power states. When both active, the MCU and the eFPGA operate at the same supply voltage, ranging between 0.5 and 0.8 V (*eFPGA On*). However, to minimize the power overhead, the eFPGA can be switched textscoff when applications do not require it (*eFPGA Off*).

The eFPGA has been implemented with conventional-well transistors, which allows reducing the leakage power while preserving the configuration during state-retentive deep sleep states by applying RBB from the external power manager (*eFPGA Retentive*). With respect to the *eFPGA Off* power state, the bitstream is kept in the eFPGA. When the application needs the eFPGA, the external power manager is asked to set the eFPGA supply voltage equal to MCU one and to set the RBB to zero. This strategy avoids costly reprogramming operations that would be needed if the eFPGA is switched textscoff. On the other hand, the MCU has been implemented in flip-well transistors. Thus, forward body-biasing (FBB) is applied to the CPU, memory, and the rest of the logic to increase performance [34], [58].

The different transistors' flavor allows reducing the power overhead on the large-area eFPGA when applications do not need it. In contrast, it allows the MCU to achieve higher performance in applications that have tight constraints and must run fast to completion.

The proposed chip relies on an external power manager. The MCU configures its power modes through SPI to enable dynamic voltage-frequency scaling (DVFS) and body-biasing policies. Future version of the SoC could make an on-chip body-bias generator [59] and power converters [9]. Fig. 3 shows how the *Arnold* chip is connected to an external power manager, the supply voltages of the main components, and the body-biasing pins.

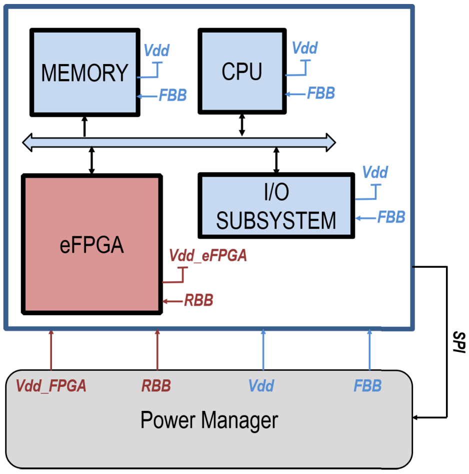


Fig. 3. *Arnold* power modes. Supply voltages and body-bias pins are driven by an external power manager.

IV. 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, and the user may specify that the signal is connected to the third 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, and so on. 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 for interrupt" (WFI) RISC-V instruction to clock-gate the CPU to save dynamic power. The compiler used for software is based on the GNU GCC 7.1.1 version, which includes the custom extensions described in [53].

V. ARNOLD PHYSICAL DESIGN

The proposed SoC fabricated in GF22FDX 10 Metal technology occupies 3 × 3 mm^2. 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 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 the performance across the process, voltage, and temperature variations.

The die picture and floorplan of the chip are shown in Fig. 4. 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 toward 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 toward the μ DMA, 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 5.11 mm^2, of which the eFPGA macro occupies 78% (4 mm^2) and the MCU 22% ((1.11 mm^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 III shows the area distribution of the chip.

The GF22FDX technology supports body biasing to modulate the tradeoff between operating frequency and leakage power dynamically. As opposed to traditional bulk technologies, due to the buried oxide providing dielectric isolation of the source and drain, the back-bias voltage can be varied over a wide voltage range from −1.8 V using conventional-well transistors adopted in the eFPGA subsystem to +1.8 V flip-well transistors adopted in the CPU subsystem. RBB is applied from an external source to minimize the eFPGA leakage power overheads while preserving the configuration. On the other hand, FBB is applied to the CPU, memories, and the rest of the logic to increase performance.

A. Performance and Energy Efficiency

In this section, 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. 5 (left) shows the maximum frequency, power consumption, and power density 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 a 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 due 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 and 0.8 V, respectively. Fig. 5(g) shows the effect of the FBB on the MCU power consumption, and Fig. 5(h) shows the effect of the FBB 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 [60].

TABLE III
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%

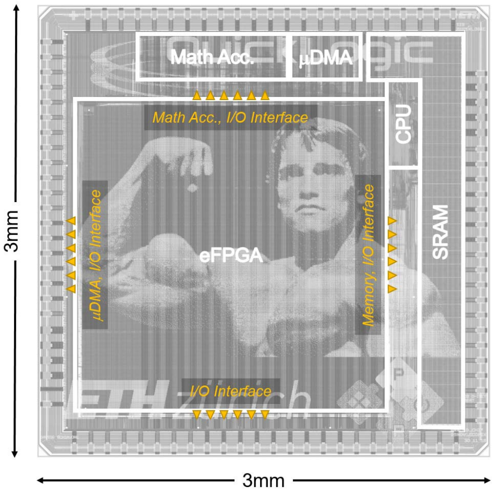


Fig. 4. Die photograph of the proposed design with the main components and eFPGA pins highlighted.

Fig. 5 (center) shows the eFPGA measured results. Fig. 5(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.

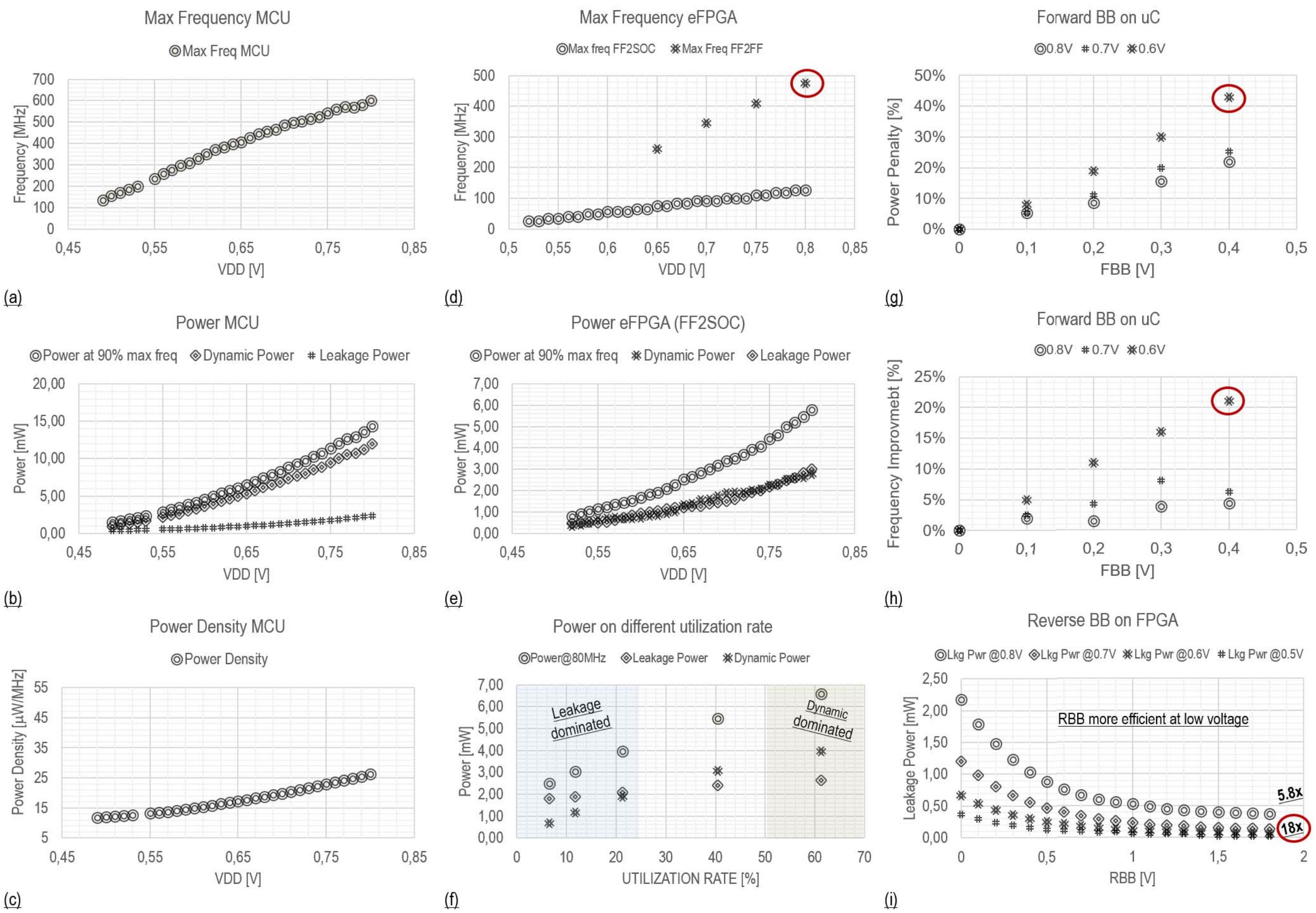


Fig. 5. (a) Frequency, (b) power consumption, and (c) energy efficiency with respect to the supply voltage of the MCU part of the proposed design. In the center, (d) frequency and power of the eFPGA macro with respect to (e) supply voltage and power with respect to (f) utilization rate. The effect of the FBB on (g) power and (h) frequency on the MCU. (i) Effect of RBB on the eFPGA leakage power during state-retentive deep-sleep mode.

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 FF to FF delay, without considering the propagation and setup timing of the eFPGA and the external logic at its boundary. The output of the Q-pin of the MSB FF 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 [see Fig. 5(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- and 1.8-V RBB as shown in Fig. 5(i), i.e., a 5.8× (at 0.8 V) to 18× (at 0.5 V) reduction can be achieved due to RBB. This result makes the eFPGA power consumption significantly reduced when not used, minimizing the integration cost and overhead. Fig. 5(f) shows how the power consumption changes with respect to the utilization rate. A design with a parameterizable 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 and 26.38 MHz, respectively. This result has been measured when the eight-parallel 32-bit accumulators are mapped on the eFPGA.

VI. USE CASES

To demonstrate the flexibility and efficiency of our heterogeneous reconfigurable SoC, three different use cases have been

TABLE IV

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
I/O Accelerator	3	497	457	6.3	2.4
BNN	0	854	1229	12.5	2.2
CRC	0	20	47	7.5	42.2

implemented, highlighting the versatility offered by embedded programmable logic.

A. I/O Subsystem Accelerator

In the context of applications for biosignal processing, it is common to extract features in the frequency domain to classify activities sensed from skeletal muscles or the brain [62]. Wavelet or Fourier transforms are used to convert the signal from the time to the frequency domain, and then, features, such as 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 preprocessing 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 transform (HDWT), which is an attractive algorithm to implement in an eFPGA as it does not require multipliers [63].

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 [64]. 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 eight 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. As the execution time is dominated by the acquisition of the SPI packets, the eFPGA-based accelerator is 2.4× more energy efficient than the CPU in both the applications. The design utilizes 20% of the available SLCs, and it uses a memory interface port, the APB interface, and four GPIOs (three output pins and one input pin), and it generates one event.

B. 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, and so on. 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 [65] or event-based audition sensors [66]. Another example where FPGAs are used to control and transfer data are bridges for off-chip accelerators (see [67] or [68]). In this context, to illustrate the flexibility of the MCU+eFPGA combination, a controller for the systolic long short-term memory recurrent neural network (LSTM-RNN) accelerator presented in [67] 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. 6 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 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.

C. CPU Subsystem Accelerator

In the context of on-the-edge computation, accelerators are used to increase performance and the energy efficiency of such devices [69]. For pattern recognition tasks in the visual domain, deep quantized neural networks are an attractive model due to their limited memory and computational requirements [70], that in extreme cases, use single-bit representation for weights and data, leveraging simple operations as logic XOR. Such neural networks are called binary neural networks (BNNs) [71]. The eFPGA has sufficient resources to allow these accelerators to be implemented, freeing the core for other computing tasks.

TABLE V

PERFORMANCE COMPARISON WITH STATE-OF-THE-ART MCU AND eFPGA SYSTEMS

	Borgatti [29]	**Lodi** [30]	**Renzini** [31]	**Fournaris** [52]	**Whatmough** [46]	**Bol** [11]	**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 AHB, 64 bit RX DMA	GPIOs AHB, 32 bit TX/RX DMA	APB, 32 bit	GPIOs AHB, 64 bit TX/RX DMA	AXI, 128 bit	- - -	GPIOs crossbar, 128 bit TX/RX DMA
FPGA Lkg Pwr *	-	-	-	-	12000 [7]	-	20.5 - 2178
FPGA Max Freq.**	175	166	50	160	734	-	475
FPGA Pwr Density ***	-	-	34.72@1.2V [8]	962@1.2V [9]	-	-	31.98@0.6V [10]
MCU Lkg Pwr *	-	-	-	7000[11]	-	1 - 30	532 - 2386
MCU Max Freq. **	175	166	50	166	-	80	600
MCU Pwr Density***	-	-	101.22@1.2V[8]	31@1.2V[12]	-	3@0.4V, 48MHz	11.88@0.49V, 135MHz
MCU+FPGA Pwr Density ***	-	1807.23@1.8V	135.94@1.2V [8]	993@1.2V [9,12]	-	-	46.83@0.52V [13]
FPGA Lkg Control	No	No	No	No	No	-	RBB

* Power numbers are in μW ** Frequency numbers are in MHz *** Power density numbers are in μW/MHz [1] Two 64 kB of L1 cache 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 [61]. [4] 6304 × 2 flip-flops for two logic tile. 5024 × 2 for the two DSP tiles [61]. [5] Signed multiplication, dot product, and built-in addition, subtraction, and accumulation units. [6] 40 × 2 MACs for the two DSP tile [61]. [7] 3 mW reported in the datasheet [61]. Assuming it is for a 1 × 1 tile, [46] uses a 2 × 2 tile, thus 12 mW have been reported in the Table. [8] Average measurements. [9] Estimated from [52]. 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 [52]. 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

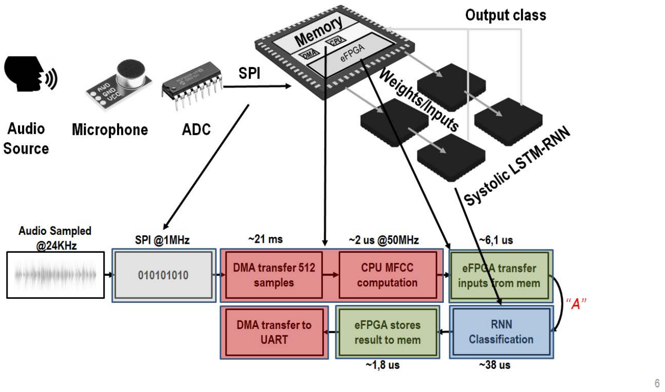


Fig. 6. Example of an application where the proposed design is driving custom protocol off-chip accelerators. Data coming from microphones are first preprocessed by the MCU and then sent to the off-chip accelerator via eFPGA for classification.

The BNN accelerator designed for this scope has four interfaces toward the main memory to maximize the bandwidth, and it is a simplified version of the accelerator presented in [12]. It assumes that input layers and filters are organized as a 3-D 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 $f0, \ldots, f7$ 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, and 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 and 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, and so on). The design occupies 42% of the SLCs available, and it uses four memory interfaces, the APB port, and it generates one event. The application consumes 12.5 mW (eFPGA+MCU), and it runs in 371 μs at 125 MHz (3 cycles/bit). Although the core implements custom instructions to speed up such kernels (as the pop count instruction), 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 (26 cyles/bit), 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 [72]. Such an accelerator uses the I/O DMA interface to leverage the linear address generator already present in the μDMA, 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 one interface toward the μ DMA with configuration, TX/RX ports. The application consumes only 7.5 mW (eFPGA+MCU), and it runs in 3.7 μs at 193 MHz (0.7 cycles/byte) for 1024-byte data, processing 4 bytes at the same time. The CPU consumes 15 mW, and it runs in 78 μs (45.6 cycles/byte) 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 [73], 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 FFs.

Table IV shows the number of GPIOs, number of FFs, 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 such as 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 V 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 [11] has a 4× smaller power density (μW/MHz). However, our MCU features 8× larger memory capacity and significantly larger peak performance as well: 7.5× higher maximum frequency, 3.19 versus 2.33 Coremark/MHz, and almost 6× better performance in near-sensor processing workloads when compared to the ARM Cortex-M0 processor used in [11]. Hence, our energy efficiency on the targeted application domain is 1.5× better.

The advanced MCU+eFPGA system presented in [46] 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 FFs, and provides remarkable energy efficiency of 312 GOPS/W. Due 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 RBB 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 [46]. The Microsemi SmartFusion2 SoC [28] used in [52] 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 *et al.* [29] and Lodi *et al.* [30] 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 [29]–[31], 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 RBB, paving the way for flexible fully programmable IoT end nodes.

VII. Conclusion

In this article, we presented *Arnold*, an RISC-V-based MCU extended with an eFPGA 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, 16 events to interact with the CPU, GPIOs, and APB. This article 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 FFs, 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. This article shows that integrating an eFPGA in an MCU in GF22FDX gives IoT devices the high versatility needed for extended product life and shorter time-to-market, still without waiving performance, power, and energy efficiency.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 13–16.

[2] A. A. Shkel and E. S. Kim, “Continuous health monitoring with resonant-microphone-array-based wearable stethoscope,” *IEEE Sensors J.*, vol. 19, no. 12, pp. 4629–4638, Jun. 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 Things J.*, 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 J.*, vol. 19, no. 13, pp. 5219–5228, Jul. 2019.

[5] A. Casson, D. Yates, S. Smith, J. Duncan, and E. Rodriguez-Villegas, “Wearable electroencephalography,” *IEEE Eng. Med. Biol. Mag.*, 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 Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 9, pp. 1990–2003, Sep. 2014.

[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,” *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[8] D. Rossi *et al.*, “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 J. Solid-State Circuits*, vol. 54, no. 7, pp. 1970–1981, Jul. 2019.

[10] D. Bol *et al.*, “Sleepwalker: A 25-MHz 0.4-V sub mm^2 7-μW/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.

[11] D. Bol *et al.*, "19.6 A 40-to-80 MHz sub-4μW/MHz ULV cortex-M0 MCU SoC in 28 nm FDSOI with dual-loop adaptive back-bias generator for 20μs wake-up from deep fully retentive sleep mode," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 322–324.

[12] 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 Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2940–2951, Nov. 2018.

[13] R. Bansal and A. Karmakar, "Closely-coupled lifting hardware for efficient DWT computation in an SoC," *J. Signal Process. Syst.*, vol. 92, pp. 225–237, Jul. 2019.

[14] 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 Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 530–543, Feb. 2020.

[15] T. Fritzmann, U. Sharif, D. Muller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum co-processors based on RISC-V," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1148–1153.

[16] D. Neil and S.-C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2621–2628, Dec. 2014.

[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 Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 1, pp. 274–287, Jan. 2019.

[18] X. Yu *et al.*, "A data-center FPGA acceleration platform for convolutional neural networks," in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2019, pp. 151–158.

[19] H. Chen, S. Madaminov, M. Ferdman, and P. Milder, "Sorting large data sets with FPGA-accelerated samplesort," in *Proc. IEEE 27th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM)*, Apr. 2019, p. 326.

[20] Z. U. A. Khan and M. Benaissa, "High-speed and low-latency ECC processor implementation Over GF(2^m) on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 165–176, Jan. 2017.

[21] J. Liao *et al.*, "FPGA implementation of a Kalman-based motion estimator for levitated nanoparticles," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 7, pp. 2374–2386, Jul. 2019.

[22] H. Homulle, S. Visser, and E. Charbon, "A cryogenic 1 GSa/s, soft-core FPGA ADC for quantum computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 11, pp. 1854–1865, Nov. 2016.

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

[24] P. Anagnostou *et al.*, "Torpor: A power-aware HW scheduler for energy harvesting IoT SoCs," in *Proc. 28th Int. Symp. Power Timing Modeling, Optim. Simulation (PATMOS)*, Jul. 2018, pp. 54–61.

[25] V. Rosello, J. Portilla, and T. Riesgo, "Ultra low power FPGA-based architecture for wake-up radio in wireless sensor networks," in *Proc. 37th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Nov. 2011, pp. 3826–3831.

[26] 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 *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2015, pp. 1–5.

[27] *Xilinx: Zynq-7000 SoC*. Xilinx ds190-Zynq-7000 datasheet. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[28] *SmartFusion: SmartFusion2 SoC*. Accessed: Feb. 17, 2021. [Online]. Available: https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2

[29] M. Borgatti, F. Lertora, B. Foret, and L. Cali, "A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable I/O," *IEEE J. Solid-State Circuits*, vol. 38, no. 3, pp. 521–529, Mar. 2003.

[30] A. Lodi *et al.*, "XiSystem: A XiRisc-based SoC with reconfigurable IO module," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 85–96, Jan. 2006.

[31] F. Renzini, C. Mucci, D. Rossi, E. F. Scarselli, and R. Canegallo, "A fully programmable eFPGA-augmented SoC for smart power applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 489–501, Feb. 2020.

[32] *NXP: Power Consumption and Measurement of i.MXRT1050*. Accessed: Feb. 17, 2021. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN12094.pdf

[33] *STMicroelectronics: STM32L476xx Datasheet*. Accessed: Feb. 17, 2021. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32l476je.pdf

[34] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, "Quentin: An ultra-low-power PULPissimo SoC in 22 nm FDX," in *Proc. IEEE SOI-3D-Subthreshold Microelectron. Technol. Unified Conf. (S3S)*, Oct. 2018, pp. 1–3.

[35] *Microsemi IGLOO Nano Low Power Flash FPGAs Datasheet*. Accessed: Feb. 17, 2021. [Online]. Available: https://www.microsemi.com

[36] *Lattice Semiconductor iCE40 UltraLite Family Data Sheet*. Accessed: Feb. 17, 2021. [Online]. Available: http://www.latticesemi.com

[37] *Menta: eFPGA IP Cores*. Accessed: Feb. 17, 2021. [Online]. Available: https://www.menta-efpga.com/efpga-ips

[38] Speedcore FPGA Datasheet. *Achronix Speedcore eFPGA*. [Online]. Available: https://www.achronix.com/sites/default/files/docs/Speedcore_Gen4_eFPGA_Datasheet_DS012.pdf

[39] *Flex-Logix eFPGA*. [Online]. Available: https://flex-logix.com/efpga/

[40] *Quicklogic: ArcticPro 2 eFPGA*. Accessed: Feb. 17, 2021. [Online]. Available: https://www.quicklogic.com/products/efpga/arcticpro-2/

[41] *NXP: I.MX 7ULP Applications Processor Consumer Products*, Datasheet, NXP Semicond., Eindhoven, The Netherlands, 2019.

[42] *Silicon Labs: EFM32 Giant Gecko 11 32bit Microcontrollers*, Datasheet, Silicon Labs Inc., Austin, TX, USA, 2019.

[43] *Xilinx: Virtex Ultrascale+*. [Online]. Available: https://www.xilinx.com

[44] GAP8 Product Brief. *GreenWaves Technology: GAP8 PRODUCT BRIEF*. [Online]. Available: https://gwt-website-files.s3.eu-central-1.amazonaws.com/Product+Brief+GAP8+-+V1_9.pdf

[45] 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.

[46] P. N. Whatmough *et al.*, "A 16 nm 25 mm^2 SoC with a 54.5x flexibility-efficiency range from dual-core arm cortex-A53 to eFPGA and cache-coherent accelerators," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C34–C35.

[47] P. Davide 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 (PATMOS)*, Sep. 2017, pp. 1–8.

[48] B. Pandey *et al.*, "Performance evaluation of FIR filter after implementation on different FPGA and SOC and its utilization in communication and network," *Wireless Pers. Commun.*, vol. 95, no. 2, pp. 375–389, Jul. 2017.

[49] W. Qiao, Z. Fang, M.-C.-F. Chang, and J. Cong, "An FPGA-based BWT accelerator for Bzip2 data compression," in *Proc. IEEE 27th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM)*, Apr. 2019, pp. 96–99.

[50] A. Di Mauro, F. Conti, and L. Benini, "An ultra-low power address-event sensor interface for energy-proportional Time-to-Information extraction," in *Proc. 54th Annu. Design Autom. Conf.*, Jun. 2017, pp. 1–6.

[51] T. Gomes, S. Pinto, T. Gomes, A. Tavares, and J. Cabral, "Towards an FPGA-based edge device for the Internet of Things," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2015, pp. 1–4.

[52] 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.

[53] 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.

[54] A. Waterman *et al.*, "The RISC-V instruction set manual," RISC-V Int. Assoc., Zürich, Switzerland, Tech. Rep., 2019, vol. 1.

[55] A. Rahimi, I. Loi, M. R. Kakoee, and L. Benini, "A fully-synthesizable single-cycle interconnection network for shared-L1 processor clusters," in *Proc. Design, Autom. Test Eur.*, Mar. 2011, pp. 1–6.

[56] A. Pullini, D. Rossi, G. Haugou, and L. Benini, "μDMA: An autonomous I/O subsystem for IoT end-nodes," in *Proc. 27th Int. Symp. Power Timing Modeling, Optim. Simulation (PATMOS)*, Sep. 2017, pp. 1–8.

[57] D. E. Bellasi and L. Benini, "Smart energy-efficient clock synthesizer for duty-cycled sensor SoCs in 65 nm/28 nm CMOS," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2322–2333, Sep. 2017.

[58] A. D. Mauro, F. Conti, P. D. Schiavone, D. Rossi, and L. Benini, "Pushing on-chip memories beyond reliability boundaries in micropower machine learning applications," in *IEDM Tech. Dig.*, Dec. 2019, p. 30.

[59] F. Zaruba, F. Schuiki, S. Mach, and L. Benini, "The floating point trinity: A multi-modal approach to extreme energy-efficiency and performance," in *Proc. 26th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Nov. 2019.

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

[61] *EFLX: EFLX 4K Product Brief for TSMC 12FFC+/12FFC/16FFC+/FFC/FF+*. Accessed: Feb. 17, 2021. [Online]. Available: https://flex-logix.com/efpga/

[62] 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,” *Inf. Fusion*, vol. 43, pp. 66–76, Sep. 2018.

[63] F. H. Elfouly, M. I. Mahmoud, M. I. Dessouky, and S. Deyab, “Comparison between Haar and Daubechies wavelet transformations on FPGA technology,” *Int. J. Comput., Inf., Syst. Sci., Eng.*, vol. 2, no. 1, pp. 96–101, 2008.

[64] 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 *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 752–757.

[65] Inivation. (2020). *Inivation Dynamic Vision Platform*. [Online]. Available: https://inivation.com/dvp/

[66] S. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, “Asynchronous binaural spatial audition sensor with 2×64×4 channel output,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 453–464, Aug. 2014.

[67] F. Conti, L. Cavigelli, G. Paulin, I. Susmelj, and L. Benini, “Chipmunk: A systolically scalable 0.9 mm^2, 3.08Gop/s/mW @ 1.2 mW accelerator for near-sensor recurrent neural network inference,” in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2018, pp. 1–4.

[68] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, “UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision,” *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

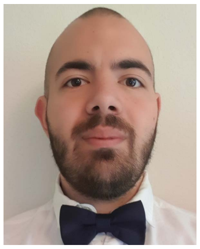
[69] D. Chen, J. Cong, S. Gurumani, W. Hwu, K. Rupnow, and Z. Zhang, “Platform choices and design demands for IoT platforms: Cost, power, and performance tradeoffs,” *IET Cyber-Phys. Syst.: Theory Appl.*, vol. 1, no. 1, pp. 70–77, Dec. 2016.

[70] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Quantized neural networks: Training neural networks with low precision weights and activations,” *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[71] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[72] E. Tsimbalo, X. Fafoutis, and R. J. Piechocki, “CRC error correction in IoT applications,” *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 361–369, Feb. 2017.

[73] *PolarFire SoC Advance Product Overview*. Accessed: Feb. 17, 2021. [Online]. Available: https://www.microsemi.com/product-directory/soc-fpgas/5498-polarfire-soc-fpga#resources



Pasquale Davide Schiavone received the B.Sc. and M.Sc. degrees in computer engineering from the Polytechnic of Turin, Turin, Italy, in 2013 and 2016, respectively. He is currently working toward the Ph.D. degree at the Integrated Systems Laboratory, ETH Zürich, Zürich, Switzerland.

In 2018, he was a Ph.D. Visiting Student with the Centre for Bio-Inspired Technology, Imperial College London, London, U.K. His research interests include datapath blocks design, low-power microprocessors in multicore systems and deep-learning architectures for energy-efficient systems.

Davide Rossi (Member, IEEE) 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 currently holds an assistant professor position. His research interests focus on energy-efficient digital architectures in the domain of heterogeneous and reconfigurable multicore and many-core systems on a chip.

Alfio Di Mauro received the M.Sc. degree in electronic engineering from the Electronics and Telecommunications Department (DET), Politecnico di Torino, Turin, Italy, in 2016. He is currently working toward the Ph.D. degree in electrical engineering at the Integrated Systems Laboratory, ETH Zürich, Zürich, Switzerland.

In January 2017, he started to work as a Researcher Assistant at the Integrated System Laboratory (IIS), Swiss Federal Institute of Technology of Zurich, Zürich, Switzerland, in the group led by Prof. Luca Benini. His research is mainly focused on the design of digital ultralow-power (ULP) system-on-chip (SoC) for event-driven edge computing.

Frank K. Gürkaynak received the B.Sc. and M.Sc. degrees in electrical engineering from the Istanbul Technical University, Istanbul, Turkey, and the Ph.D. degree in electrical engineering from ETH Zürich, Zürich, Switzerland, in 2006.

He is currently working as a Senior Researcher at the Integrated Systems Laboratory, ETH Zürich. His research interests include digital low-power design and cryptographic hardware.

Timothy Saxe received the B.S.E.E. degree from North Carolina State University, Raleigh, NC, USA, in 1975, and the M.S.E.E. degree and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1976 and 1980, respectively.

He joined QuickLogic Corporation, San Jose, CA, USA, in May 2001. He has held a variety of executive leadership positions at QuickLogic, including the Vice President of Engineering and the Vice President of Software Engineering. He has served as our Senior Vice President of Engineering and the Chief Technology Officer since August 2016 and the Senior Vice President and the Chief Technology Officer since November 2008.

Mao Wang received the B.S. degree in electrical engineering and the M.S. degree in engineering management from Santa Clara University Santa Clara, CA, USA, in 1998 and 2000, respectively.

He is currently the Senior Director of Product at QuickLogic Corporation, San Jose, CA, USA, with the mission of democratizing embedded field-programmable gate array (FPGA) into every system-on-chip (SoC).

Ket Chong Yap received the B.S. degree in electrical engineering from Iowa State University, Ames, IA, USA, in 1990.

He joined QuickLogic Corporation, San Jose, CA, USA, in September 1999, actively participating in QuickLogic field-programmable gate array (FPGA) product development. Prior to joining QuickLogic, he was with EXEL Microelectronics, San Jose, CA, USA, as a Quality Assurance Engineer from 1990 to 1991, a Product/Test Engineer from 1992 to 1994, and a Design Engineer from 1995 to 1996, working on EEPROM technology. He was involved with Programmable Microelectronics Corporation, San Jose, CA, USA, from 1997 to 1998, working on FLASH memory.

Luca Benini (Fellow, IEEE) received the Ph.D. degree from Stanford University, Stanford, CA, USA, in 1997.

He served as a Chief Architect at STmicroelectronics, Grenoble, France. He holds the Chair of Digital Circuits and Systems, ETH Zürich, Zürich, Switzerland, and is currently a Full Professor with the Universita di Bologna, Bologna, Italy. He has published more than 1000 peer-reviewed articles and five books. His research interests are in energy-efficient parallel computing systems, smart sensing microsystems, and machine learning hardware.

Dr. Benini is a Fellow of the Association for Computing Machinery and a member of the Academia Europaea. He was a recipient of the 2016 IEEE CAS Mac Van Valkenburg Award.