A Fully Programmable eFPGA-Augmented SoC for Smart Power Applications

Francesco Renzini, Claudio Mucci, Davide Rossi, Eleonora Franchi Scarselli, *Member, IEEE*, and Roberto Canegallo

***Abstract*— This paper proposes a reconfigurable system on chip (SoC) for smart power applications. The system is composed of an ultra-low-power microcontroller for standard software programmability, coupled to an embedded-FPGA (eFPGA) to perform control-driven applications and lightweight digital signal processing, at lower power consumption and higher responsiveness than with processor-based execution. To the best of our knowledge, this is the first heterogeneous reconfigurable SoC targeting smart power applications. The SoC targets BCD technologies integrating bipolar, CMOS, and DMOS devices, typically featuring a small number of metal layers when compared with the traditional CMOS technologies. The added value of the proposed system is that the digital system is fully synthesizable since the eFPGA is based on a soft-core approach. This paper presents the results of integrating an eFPGA with a computational capability of ≃1k equivalent gates in STMicroelectronics 90-nm BCD technology featuring five metal layers and high-*k* transistors. We benchmarked our architecture on a wide range of applications relevant to the smart power domain. eFPGA integration in SoCs introduces a 20%–27% area overhead but has a straightforward benefit in terms of energy consumption, which proves reduction from about 10× to 800×. In terms of latency, the eFPGA implementation allows a gain from 8× to 145× comparing the pure cycles count.**

***Index Terms*— Embedded FPGA, microcontroller, SoC, smart power.**

I. Introduction

An increasing amount of applications, often referred to as the “Internet of Things” (IoT), require deeply embedded intelligent systems (a.k.a. the end-nodes of the IoT) to connect to each other and interact with the real-world (cyber-physical interaction). This paradigm requires systems capable of sensing the environment, elaborating the data acquired, transmitting compressed information and - last but not least - providing feedback to physical objects [1].

The control and processing unit of the end-nodes can be either software-programmable, hardware-programmable or hardwired in an Application Specific Integrated Circuit (ASIC) as well as a combination of these solutions, in a more general scenario. Software-programmability is the most flexible option, as it allows one to employ microprocessor (μP) or Digital Signal Processors (DSPs) which execute a software program, implementing acquisition, processing, and wireless stacks. The hardware-programmable devices are Programmable Logic Devices (PLDs), nowadays led by Field Programmable Gate Arrays (FPGAs). This kind of device adapts its configurable architecture, hence its functionality, based on applications requirements and is typically more powerful and efficient than software-based solutions for compute intensive tasks. On the other hand, the development time for FPGA applications is much longer than the software one. ASICs typically provide the best performance in terms of power consumption and area occupation; however their hardwired functionalities do not allow these systems to be reused for different application domains, entailing significant Non-Recurring Engineering (NRE) costs, which somehow limits their utilization in generic IoT applications.

Microcontrollers (μC) integrate microprocessor and/or DSP architectures with some peripherals, which are useful to perform the physical world interaction, such as General Purpose I/Os (GPIOs), Pulse Width Modulation (PWM) controllers, analog-to-digital and digital-to-analog converters. System-on-Chips (SoCs) which implement both microcontroller and embedded FPGA (eFPGA) have been proposed to achieve a good compromise in terms of flexibility and performance. Software-programmable devices are probably the most commonly used digital controllers, thanks to their computation unit, useful configurable peripherals and relatively low cost. Since control application requirements have become more stringent from the viewpoint of processing time and the number of I/Os, many designers have opted for a combination of both software-programmable and hardware-programmable devices [2]. In addition, software-programmable devices are not convenient to perform compute-intensive kernels due to their inherent sequential mode of operation, whereas hardware-programmable devices - thanks to their fixed parallel structure - enable one to improve the efficiency and timing performance - switching frequency - [3]–[5]. On the other hand, advanced FPGAs such as the ones commercialized by Xilinx or Intel are significantly more expensive than MCUs, hence not suitable for an IoT scenario where smart power systems are massively deployed.

Manuscript received April 1, 2019; revised June 7, 2019; accepted July 16, 2019. Date of publication August 7, 2019; date of current version January 31, 2020. This work was supported in part by the H2020-EU ECSEL 737417: R3-PowerUP 300mm Pilot Line for Smart Power and Power Discrete. This paper was recommended by Associate Editor Y. Lu. *(Corresponding author: Francesco Renzini.)*

F. Renzini, D. Rossi, and E. Franchi Scarselli are with ARCES–DEI, University of Bologna, I-40126 Bologna, Italy (e-mail: francesco.renzini@unibo.it; davide.rossi@unibo.it; eleonora.franchi@unibo.it).

C. Mucci and R. Canegallo are with STMicroelectronics, I-20864 Agrate Brianza, Italy (e-mail: claudio.mucci@st.com; roberto.canegallo@st.com).

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

Digital Object Identifier 10.1109/TCSI.2019.2930412

TABLE I
PROGRAMMABLE DEVICES SCENARIO

		Applications	Technology	Metal Layers	Temp. Range [°C]	IP	Area [mm^2]	embedded MCU/FPGA	eq. gates
μC	STM32 [14]	Gen. Purpose	CMOS	n.a.	-40–125	soft	medium	✓/✗	n.a.
	PULPino[a][15]	Gen. Purpose	CMOS 65 nm	9	-40–125[b]	soft	medium	✓/✗	n.a.
FPGA	Xilinx [16]	Gen. Purpose	Adv. CMOS	> 10	-55–125	hard	huge	✗/✓	$\leq$ M
	Intel [17]	Gen. Purpose	Adv. CMOS	> 10	-40–130	hard	huge	✗/✓	$\leq$ M
	QuickLogic [18]	Gen. Purpose	Adv. CMOS	> 10	-55–125	hard	huge	✗/✓	$<$ M
μFPGA	iCE40 [19]	Rec. I/Os/Accel.	CMOS 40 nm	n.a.	-55–125	hard	small	✗/✓	0.4–8 k
	IGLOO2 [20]	Rec. I/Os/Accel.	CMOS 65 nm	n.a.	-55–125	hard	small	✗/✓	$<$ 150 k
eFPGA	[18], [21], [22], [23]	Accelerator	Adv. CMOS	n.a.	-40–125	hard	big	✗/✓	$<$ M
	Kim2017 [24]	Accelerator	CMOS 65 nm	n.a.	n.a.	soft	n.a.	✗/✓	n.a.
	Cuppini2015 [25]	Accelerator	CMOS 65 nm	7	-40–125	soft	0.4/0.2[c]	✗/✓	1 k
	Cuppini2015 [25]	Smart Power	BCD 110 nm	4	-40–150	soft	1.2/0.7[c]	✗/✓	1 k
SoC	Borgatti2003 [26]	Reconf. I/Os	CMOS 180 nm	6	n.a.	hard	20 (8.2)[d]	✓/✓	15 k
	XiSystem [27]	Periph/Acceler	CMOS 130 nm	6	-40–125[b]	hard	42 (6)[d]	✓/✓	15 k
	Morpheus [28]	Periph/Acceler	CMOS 90 nm	7	-40–125[b]	hard	110	✓/✓	15–100 k
	[16], [17]	Gen. Purpose	Adv. CMOS	> 10	n.a.	hard	huge	✓/✓	$\leq$ M
	Proposed SoC	Smart Power	BCD 90 nm	5	-40–150	soft	1.78 (0.347)[d]	✓/✓	1 k

[a] Imperio implementation (http://asic.ethz.ch/2015/Imperio.html)
[b] Personal communication
[c] Max speed/Min area implementations
[d] System area (eFPGA area)

In this paper we focus on the smart power area where the end-nodes need both to manage simple control policies and to interact with the real world - e.g. motion and lighting control [6] [7]. Among the numerous definitions of smart power, the one presented in [8] is the coexistence of both "force" - power electronic - and "intelligence" - CMOS circuits - in the same chip. Compared to current commercial off-the-shelf microcontroller systems, smart power technologies allow on-chip integration of power devices which are typically hosted on PCBs or Systems-in-Package in traditional systems, with a significant benefit in terms of miniaturization and cost. On top of that, for an IoT scenario, Ultra Low-Power (ULP) methods (e.g. sleep walking, power gating) can also be applied to this kind of system to keep energy/power consumption under control. In particular, BCD technology allows one to integrate in a single die Bipolar, CMOS and DMOS transistors for power circuits [9]. BCD technology has proven to be suitable for sensors [10] and actuators [11]. Traditionally, the systems realized in BCD technologies were implemented as ASICs due to the less scaled nature of this technology with respect to traditional CMOS and to the limited amount of metal layers, which increase the overhead of the routing. However, the scaling of BCD's CMOS transistors, down to 90 nm and beyond, makes this technology suitable for integration of mid-complexity digital circuits like μCs. This allows one to enable BCD utilization in the IoT arena [12], which requires additional features such as wireless stack implementation, radio interfaces, etc.

To the best of the authors' knowledge, in this paper we present for the first time a heterogeneous smart power digital core system integrating a microcontroller coupled with an embedded FPGA. The whole digital core is fully-synthesizable in standard cell libraries, in order to be both portable to different technology and adaptable to a preexisting floorplan - as with smart power BCD technology where area occupation is dominated by power transistors. The digital core presented in [13] - where an implementation in 130 nm BCD technology was discussed - is implemented in 90 nm BCD technology featuring 5 metal layers. The eFPGA is tailored for simple controllers, hence it has small computational capability - about 1k equivalent-gates. We evaluate several applications relevant to a wide range of connected smart power devices and we compare the eFPGA performance with respect to open-source embedded microcontroller, commercially available microcontroller and ASIC solutions. In the energy-aware analysis we demonstrate the energy efficiency of the eFPGA in handling different kinds of smart power applications, achieving a $10\times$–$800\times$ energy gain over microcontroller architecture. Hence, the eFPGA can manage smart power tasks, while the microcontroller can handle other computations such as high-level communication stack or data processing or if this is not required it can be switched to sleep-mode to reduce power consumption.

The paper is organized as follows: Section II provides a reminder of related works. In Section III we present the proposed digital core architecture, while the implementation results are described in Section IV. The energy efficiency and computational base model are described in Section V. An applications analysis and relative results are given respectively in Sections VI and VII. The conclusions are summarized in Section VIII.

II. RELATED WORK

The programmable devices scenario is summarized in Table I, including all the solutions from standard μC to advanced SoC augmented by eFPGAs which allow one to

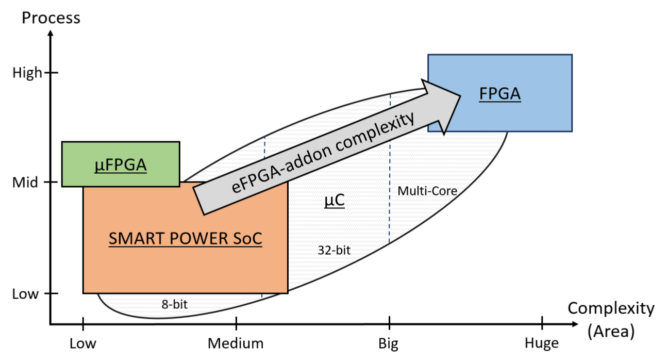


Fig. 1. Programmable devices scenario.

implement digital control. In the remainder of this section an analysis of those solutions will be presented. In Table I "eq. gates" is the computational capability of the reconfigurable device, while "soft/hard IP" means that the device is either synthesizable or based on a hard-macro approach. Fig. 1 shows an analysis of manufacturing costs versus system complexity with the different solutions.

A. Microcontrollers

There exists a huge variety of microcontrollers with different peripherals and for different applications. Every μC has almost a complete set of peripherals such as configurable PWM controllers, timers, GPIOs, etc. In Table I we refer to a commercial μC [14] and to an open-source solution [15]. The main idea is to move the computational load for a specific application from the processor to a more efficient peripheral/IP connected to the microcontroller bus. Depending on the application requirements there exists a huge variety of microcontrollers equipped, for instance, with a Floating Point Unit (FPU), LCD controller, camera controller, etc. [29]. Microcontrollers have also been developed for example with either ultrasonic sensing peripherals [30], brushless motor controller [31] or radio communication interfaces [32]. Microcontroller complexity goes from an 8-bit core to a high-end 32-bit multi-core μC [33] tailored for high-performance computing, hence manufacturing costs are strongly related to the circuit complexity as reported in Fig. 1. The main advantage of microcontrollers is the ease of programmability, but on the other hand they do not allow one to design an ad-hoc dedicated controller.

B. FPGAs

Nowadays FPGAs are used in a wide range of applications. They were historically used for high-density and parallel computations as well as neural network accelerators [34], but also for power electronics applications [2]. Typically, FPGAs are designed with a hard-macro approach, repeating the optimized basic block so as to realize a huge-sized IP with enormous computational capability. The more commonly used technologies are: CMOS from 130 to 40 nm, 22 nm FD-SOI or advanced CMOS technology with FinFET. These kinds of technology have a greater number of metal layers than standard technologies for both routing and power distribution and for this reason FPGAs tend to be expensive. These advanced technologies have several metal layers for routing which make the place and route easier. Nevertheless, these kinds of technology are not typically qualified for high temperatures such as 150 °C. The main drawback of FPGAs is the high manufacturing cost of producing complex devices such as shown in Fig. 1. Some companies such as [19] and [20] have designed small FPGAs (μFPGA in Table I) in non-advanced CMOS technologies in order to reduce the device costs; they are tailored for reconfigurable I/Os and simple accelerators. The FPGAs do not have an embedded microprocessor for software-programmability, hence if one needs a processor one has to implement a soft-processor in the FPGA - which is not always possible in μFPGAs due to their small dimensions. In addition, implementing a soft-processor will not guarantee high energy efficiency because the processor is mapped in programmable-hardware and not in custom circuits. Thus, FPGAs are oversized for smart power applications - as shown in Table I and Fig. 1 - and although μFPGAs are the right size, they are still without processors.

C. Embedded FPGAs

There are several semiconductor companies developing embedded FPGA IPs, such as [18], [21]–[23] to increase system flexibility based on the applications requirements. These kinds of eFPGA are large-size devices - up to about 1M equivalent gates - optimized at transistor level for speed and area density. The eFPGAs are designed by the vendors for a specific technology - from standard to advanced CMOS as reported in Table I and Fig. 1 - in order to both employ as few metal layers as possible - to reduce the manufacturing costs - and increase density to improve the yield. Typically, these large eFPGAs are designed with a hard-macro approach, but smart power SoCs are typically analog-on-top - because area occupation is dominated by both analog and power circuits - and therefore a soft-core approach is mandatory.

The commercially available eFPGAs are usually used as hardware accelerators e.g. sensor interface preprocessors for machine learning or security algorithms, etc. The eFPGAs available on the market are useful for high-density computing and they allow one to fit computing architecture to specific tasks, thus improving the performance in terms of speed and efficiency. On the other hand, there also exist synthesizable FPGAs, e.g. [24], which uses the open-source Verilog-to-Routing (VTR) tool. Nevertheless, VTR is only able to model island-style FPGAs - two dimensional arrays of logic blocks with horizontal and vertical routing channels. One also finds synthesizable eFPGAs with a datapath-oriented structure [35]. Our proposed eFPGA is fully-synthesizable - soft IP as reported in Table I - as shown in [25] and the interconnection network is based on a multi-stage switching network [36]. This type of interconnection network allows full-routability and provides sustainable area overhead in small-size devices.

D. System-on-Chips

System-on-chips are very attractive as digital controllers because they integrate both a microcontroller and an eFPGA

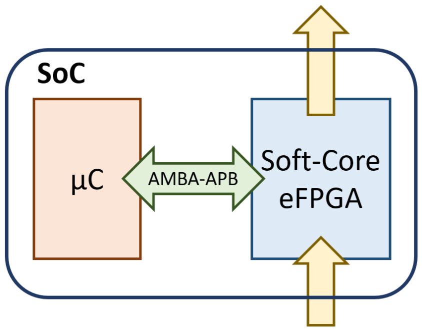


Fig. 2. Reconfigurable SoC.

in the same integrated circuit. Hence SoCs provide the best system flexibility, combining software- and hardware- programmability. In one pioneering work [26] the eFPGA is used for reconfigurable I/Os while in the systems proposed in [27] and [37] the eFPGA is adopted for custom peripherals and accelerators e.g. binarization and Ethernet MAC. As reported in Table I, these SoCs are too complex - hence expensive - for smart power applications and they are based on a hard-macro eFPGA approach just like the SoCs available on the market [16] [17], for which most of the considerations previously made for standalone FPGAs still apply.

The proposed SoC allows both software- and hardware-programmability allowing the system to be flexible. It uses a soft-core approach to adapt its floorplan for smart power devices and is realized using 5 metal layers reducing the manufacturing costs. The eFPGA has a computational complexity of 1k equivalent gates and hence is tailored for smart power applications. Thus, to the best of our knowledge the proposed SoC is the first heterogeneous integrated circuit, including both a microcontroller and an eFPGA and targeting smart power IoT applications.

III. System-Level Architecture

In this section we describe our proposed system architecture. We propose a digital circuit, as introduced in [13], which consists of a PULPino microcontroller [15] coupled with an embedded FPGA template [25] [38]. The interfacing is done through an AMBA Advanced Peripheral Bus (APB) as shown in Fig. 2. The processor manages the eFPGA like any other standard peripheral. Thus, when reading and writing registers at certain addresses defined in the RTL code, the microprocessor is able to fully-program the eFPGA. It is possible to set a frequency-divider (prescaler) to divide the system-clock-frequency for the eFPGA according to application needs. The processor can select the source and the destination of the eFPGA I/Os, which can be either primary I/Os or internal registers. The core can also reprogram the eFPGA at any time, writing the configuration bit-stream into the configuration memory of the eFPGA. The whole digital structure is soft-core, which means it is described by a synthesizable HDL code, useful for a standard cell design flow.

A. Microcontroller

PULPino is an open-source ultra-low-power microcontroller. It has a single-core processor, based on an

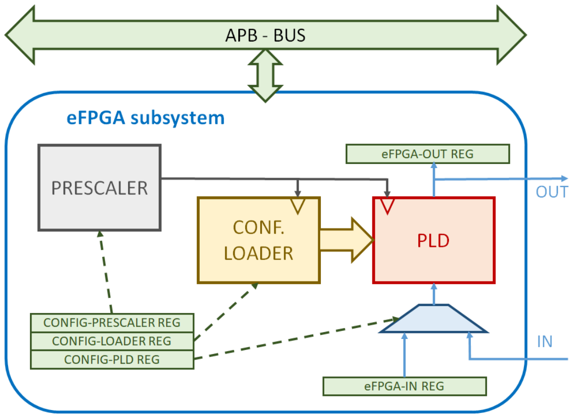


Fig. 3. The eFPGA subsystem interfaced through APB bus.

implementation of the RISC-V instruction set architecture optimized for low-power and high-energy-efficient computing. Regarding processor implementation, it is possible to use either a 32-bit 4-stage RI5CY pipeline or a 32-bit 2-stage Zero-riscy pipeline or a 32-bit 2-stage Micro-riscy pipeline without any hardware multiplier. In this work we used the 4-stage core. Since PULPino is a microcontroller, it also has traditional peripherals such as serial communication interfaces, timers, an interrupt controller and general purpose I/Os. The implemented system has 32-bit 4k-word static RAM for both instruction and data memory.

B. Embedded FPGA Sub-System

The embedded FPGA is a soft-core Intellectual Property (IP). Unlike commercial embedded FPGAs that use a hard-macro approach to optimize area occupation and performance at a transistor level, the proposed eFPGA is fully-synthesizable in standard cell libraries. Fig. 3 illustrates the eFPGA subsystem which consists of the actual Programmable Logic Device (PLD) which has 64 I/Os, a prescaler, a configuration loader and configuration registers accessible by the bus. The PULPino core addressing the CONFIG-LOADER REG register via the bus writes the configuration bit-stream into the configuration memory of the PLD thanks to the configuration loader (CONF. LOADER of Fig. 3). Depending on the application requirements, the processor can configure both the eFPGA clock frequency - programming the prescaler through the CONFIG-PRESCALER REG - and the source of the PLD inputs - writing the CONFIG-PLD REG selecting primary inputs or a register. The prescaler is used to divide the system-clock frequency by a factor n_{div} = 1–2^{16} based on both application reactivity needs and critical-path. The prescaler can also switch-off the eFPGA clock when it is not required, in order to reduce the dynamic power consumption. During task execution, the microprocessor can both program the 64 eFPGA inputs writing in eFPGA-IN REG (with 8-bit banks), and read the 64 eFPGA outputs (with 8-bit banks) to check or interact with the eFPGA - if required by the application.

1) Embedded FPGA Architecture: The soft-core of the embedded FPGA (PLD block of Fig. 3) has 64 inlets/outlets and a dimension of about 100k equivalent-gates. The PLD is provided by 16 Configurable Logic Blocks (CLBs) and the

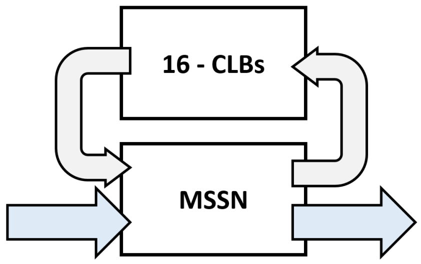


Fig. 4. Diagram of the eFPGA.

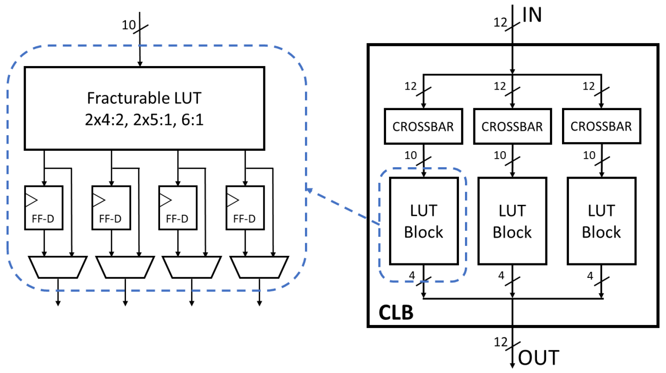


Fig. 5. eFPGA CLB structure [25].

configuration memory is made of latches replacing traditional SRAM cells so as to guarantee synthesizability. The eFPGA interconnection network is a Multi-Stage Switching Network (MSSN) with a butterfly-oriented topology (Fig. 4). This kind of network provides synthesizability and non-blocking routing features. Each CLB has 12 I/Os and 3 Basic Logic Elements (BLEs), which can be used as either 2×LookUp Table (LUT) 4:2, 2×LUT 5:1 or 1×LUT 6:1 as shown in Fig. 5. References [25] and [36] respectively describe the eFPGA structure and the MSSN characteristics. The computational capability of the eFPGA is about 1k equivalent-gates and this under exploitation of the occupied area - 100k eq. gates area occupation and 1k eq. gates of computational availability - is clearly due to the reconfigurability.

2) eFPGA Software Tools: A complete CAD flow for the proposed eFPGA was implemented as explained in [25]. The flow starts with an HDL code technology independent presynthesis in logic operators and flip-flop functionalities using Synopsys Design Compiler [39]. Then VTR provides the logic synthesis and LUT mapping and Versatile Place and Route (VPR) tool provides LUT packing and placement, while a custom tool is used to configure routing.

C. Application Programming Interface

In the proposed heterogeneous system, the task partitioning between microcontroller and eFPGA is arranged by the application designer. We developed the procedures for the PULPino code summarized in Table II. The reset_efpga function resets the eFPGA, while setup_efpga configures both the interconnection network and the CLBs writing in the CONFIG-LOADER REG of Fig. 3. The setup function programs the prescaler setting and the PLD inlets writing in both CONFIG-PRESCALER REG and CONFIG-PLD REG of Fig. 3. The microcontroller can handle eFPGA inputs by writing in the eFPGA-IN REG through the set_in_efpga function and reads the eFPGA outputs by reading the eFPGA-OUT REG with the read_out_efpga function.

TABLE II
API FUNCTION PROTOTYPES

Function Name	Description
reset_efpga()	resets the efpga
setup_efpga(efpga_addr, data_ptr)	manages configuration registers
set_in_efpga(bank_addr, value)	writes eFPGA inputs
read_out_efpga(bank_addr)	reads eFPGA outputs

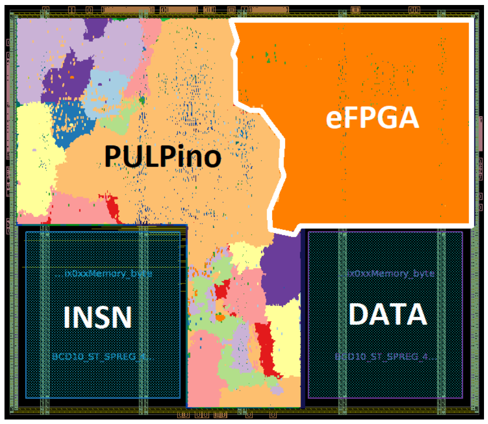


Fig. 6. Digital core BCD 90 nm implementation.

IV. Implementation Results

We implemented the digital core of the system-on-chip in STMicroelectronics BCD technology at 90 nm with 5 metal layers for the routing, using a standard cell design flow. Fig. 6 shows the whole digital core layout implemented with a target frequency of 50 MHz. The overall area occupation is 1.78 mm^2 with about 75% of row utilization - 1.3 mm^2 without considering the memory area. Fig. 6 shows two 16 kB SRAM for data and instructions - bottom side big rectangles - the soft-core eFPGA which has an area occupation of 0.347 mm^2 - and the PULPino microcontroller which includes RISC-core, standard peripherals and other control logic blocks. The eFPGA area overhead is 19.39% (26.69% without considering the memory area). The implementation results are summarized in Table III where the values in brackets do not consider the memory area.

V. Energy Efficiency Model

In this section we analyze the energy performance of the proposed architecture in carrying out various kinds of application. We evaluated the energy-efficiency of the eFPGA in managing a specific task compared to PULPino. We also consider other solutions, such as both the commercial microcontroller - STM32L152RE which integrates dedicated peripherals - and ASIC solutions - to give an idea of a dedicated solution that

TABLE III
IMPLEMENTATION RESULTS

	Implementation
Technology	BCD 90 nm
Frequency	50 MHz
System Area	1.78 (1.3) mm^2
eFPGA Area	0.347 mm^2
eFPGA Area %	19.39 (26.69) %

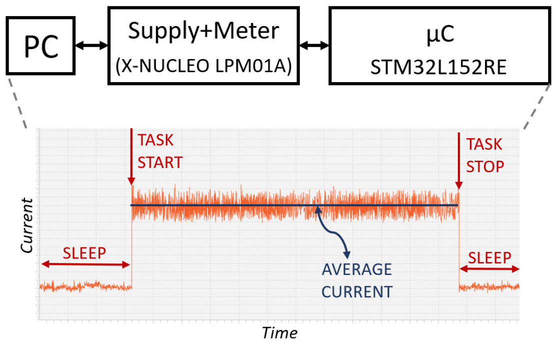


Fig. 7. Measurement setup and current profile.

has the best achievable performances - implemented in the same technology as our proposed digital core.

At the beginning we considered control applications, then streaming tasks and in the end ultra-low-power application. Through this comparison, we evaluated how energy-efficient the proposed embedded FPGA is in managing different kinds of application and in general the efficiency of the proposed digital system as a system-on-chip core. The results are summarized in Table IV and Fig. 12.

The energy data of both the PULPino-eFPGA system and the ASIC implementation were obtained after parasitic back-annotation, coming from physical synthesis. The average dynamic powers P_{eFPGA}, P_{PULP} and P_{ASIC} were estimated by annotating simulation-based switching activity using Value Change Dump (VCD) files including also Clock Tree Synthesis (CTS) power estimation. The physical synthesis flow adopted was industrially qualified by the foundry to correlate well with implementation - Section IV - and silicon. The physical synthesis was performed in STMicroelectronics BCD technology at 90 nm using Synopsys Design Compiler Graphical, while the power estimation was computed using Synopsys PrimeTime-PX in typical conditions (1.2 V and 25 °C).

For the STM32, power estimation was performed from experimental measurements. The STM32 was configured with a Multi-Speed Internal (MSI) clock in Range 3 at 524.288 kHz, with a supply voltage of 1.8 V which corresponds to a core supply voltage of 1.2 V [14]. We measured the average supply current - keeping the processor in low-power sleep mode - and then we computed the average power consumption P_{MEAS}. Fig. 7 shows the measurement setup used, mostly based on STM LPM01A which is capable of both supplying power and measuring current consumption. The STM32L1 microcontrollers are realized in 130 nm technology [40] so, to compare its energy performance with our implemented digital core (1.2 V at 90 nm), we scaled the measured power consumption following the generalized scaling theory [41] with:

$$\begin{cases} \kappa = \dfrac{VDD_{MEAS}}{VDD_{BCD}} = \dfrac{1.8\,V}{1.2\,V} \\ \lambda = \dfrac{L_{MEAS}}{L_{BCD}} = \dfrac{130\,nm}{90\,nm} \end{cases} \tag{1}$$

Hence, the power consumption comparable with BCD technology power consumption P_{STM32} is:

$$P_{STM32} = \frac{P_{MEAS}}{\lambda \cdot \kappa^2} \tag{2}$$

In order to make a right comparison between the different implementation efficiencies - eFPGA, PULPino, STM32 and ASIC - we needed to evaluate the average energy per task. For the eFPGA solution, following the model in [13], from the eFPGA average power consumption we obtained the eFPGA energy per task as power density $P_{d_{eFPGA}}$ multiplied by the number of clock cycles:

$$E_{eFPGA} = P_{d_{eFPGA}} \cdot n_{tick} = \frac{P_{eFPGA}}{f_{eFPGA}} \cdot n_{tick} \tag{3}$$

where P_{eFPGA} is the average power consumption estimated with PrimeTime-PX back-annotating switching activity from VCD file, f_{eFPGA} is the operating clock frequency of the eFPGA and n_{tick} is the number of clock cycles to execute the specific task.

The energy per task of PULPino is defined as power density $P_{d_{PULP}}$ multiplied by the number of instructions:

$$E_{PULP} = P_{d_{PULP}} \cdot n_{insn} = \frac{P_{PULP}}{f_{PULP}} \cdot n_{insn} \tag{4}$$

where P_{PULP} is the PULPino average power consumption, f_{PULP} is the clock frequency of PULPino and n_{insn} is the number of assembly instructions to execute a specific task.

For the energy per task of the second microprocessor architecture, the idea is the same as equation (4), multiplying the power density $P_{d_{STM32}}$ by the number of instructions needed for the specific task:

$$E_{STM32} = P_{d_{STM32}} \cdot n_{insn} = \frac{P_{STM32}}{f_{STM32}} \cdot n_{insn} \tag{5}$$

where P_{STM32} is the measured power consumption scaled to BCD technology using equation (2) and f_{STM32} is the STM32 clock frequency.

The ASIC implementation is similar to eFPGA - equation (3) - for energy estimation, that is: the power density $P_{d_{ASIC}}$ multiplied by the number of clock cycles n_{tick}:

$$E_{ASIC} = P_{d_{ASIC}} \cdot n_{tick} = \frac{P_{ASIC}}{f_{ASIC}} \cdot n_{tick} \tag{6}$$

where P_{ASIC} is the ASIC average power consumption, f_{ASIC} is the ASIC operating clock frequency and n_{tick} is the number of clock cycles to execute a specific task.

Considering the energy per task and not the power, it is possible to compare the performances of the different solutions. Table IV summarizes all the application results where the equivalent gates are computed comparing a 4 transistor standard cell area. Obviously, the ASIC implementation is the

TABLE IV
APPLICATIONS RESULTS

		Applications						
		PWM	RGB LED	BRUSHED	STEPPER	CRC16	LFSR	WUR
eFPGA	f_{eFPGA} [MHz]	1.25	1.25	1.25	1.25	1.25	1.25	1.25
	CLBs	7	11	7	4	4	15	5
	P_{eFPGA} [μW]	20.86	28.45	22.75	21.86	64.85	110.95	34.05
	$P_{d_{eFPGA}}$ [μW/MHz]	16.69	22.76	18.2	17.49	51.88	88.78	27.24
	n_{tick}[c]	1	1	1	1	1	1	1
	E_{eFPGA} [pJ]	16.69	22.76	18.2	17.49	51.88	88.78	27.24
PULPino	f_{PULP} [MHz]	10	n.a.[a]	10	10	10	10	n.a.[b]
	P_{PULP} [μW]	984	n.a.	1082.5	1058.4	1050	886	n.a.
	$P_{d_{PULP}}$ [μW/MHz]	98.4	n.a.	108.25	105.84	105	88.6	n.a.
	n_{insn}[c]	77	n.a.	110	145	8	42	n.a.
	E_{PULP} [pJ]	7576.8	n.a.	11907.5	15346.8	840	3721.2	n.a.
ASIC	*Area* [μm^2]	442.3	732.1	512.6	340.26	554.3	1174.4	367.7
	Eq. gates	156	260	179	125	201	442	134
	f_{ASIC} [MHz]	10	10	10	10	10	10	0.1
	P_{ASIC} [μW]	2.67	3.73	2.41	4.19	11.3	29.3	0.0398
	$P_{d_{ASIC}}$ [μW/MHz]	0.267	0.373	0.241	0.419	1.13	2.93	0.398
	n_{tick}[c]	1	1	1	1	1	1	1
	E_{ASIC} [pJ]	0.267	0.373	0.241	0.419	1.13	2.93	0.398
STM32	f_{STM32} [MHz]	0.524288	0.524288	0.524288	0.524288	0.524288	0.524288	n.a.[b]
	P_{STM32} [μW]	1.67	2.769	69.785	70.892	82.523	75.877	n.a.
	$P_{d_{STM32}}$ [μW/MHz]	3.185	5.282	133.1	135.22	157.4	144.724	n.a.
	n_{insn}[c]	sleep	sleep	79	111	10	15	n.a.
	E_{STM32} [pJ]	3.185	5.282	10515.24	15009	1574	2170.86	n.a.

[a] PULPino implementation is too inefficient
[b] μC implementations are not used for ultra-low-power applications
[c] it corresponds to latency in an isofrequency case

best feasible solution in terms of energy efficiency and area occupation - compared to Table III - even though it is not programmable.

We define the energy gain between PULPino and the eFPGA as:

$$E_{GAIN} = \frac{E_{PULP}}{E_{eFPGA}} \tag{7}$$

This model is clearly pessimistic for the eFPGA because, in our estimation E_{PULP} does not take into account processor pipeline stalls, assuming that the microprocessor executes one instruction per cycle. This ratio evaluates whether a specific task is better managed in the microprocessor or in its programmable peripheral to increase the energy efficiency of the proposed system-on-chip. Furthermore, the energy gain is useful to justify the non-negligible area overhead due to reconfigurability through adding the eFPGA, as shown in Table III.

VI. APPLICATION RESULTS

A. Control Applications

Control applications are a class of application where the computational base model is well described through a Finite State Machine (FSM). Hence, the digital controller has to generate specific outputs based on the inputs and the internal state. For this reason, control applications are event-driven applications. Typically, in this case it is necessary to manage certain physical quantities - e.g. average voltage or current based on some kinds of input - for switching regulators, motor controllers, smart-switches. As we will analyze, the eFPGA is more efficient than a processor in handling FSMs because the microprocessor is oversized for a simple FSM with few states, inlets and outlets. The whole processor pipeline also has to work to update even a few bits and it needs to access the instruction memory. The processor reads which instruction has to be executed every clock cycle. Potentially it might have to read and write-back data. Since control applications are event-driven ones, the most efficient way to implement them with a microprocessor is based on the interrupt paradigm. However, in this approach Interrupt Service Routines (ISRs) have a prologue and epilogue which significantly affect power consumption. The system latency depends on both clock frequency and the number of clock cycles to execute the task - n_{tick} and n_{insn} in Table IV. For this reason, implementations based on the hardware-paradigm, be it ASIC or FPGA-based, are more reactive than processor-based solutions, since they respond in just 1 cycle. To achieve the same latency a processor needs to increase the operating working frequency by an $\simeq n_{insn}$ factor (assuming 1 instruction per cycle) producing a straightforward drawback in terms of power. As control application examples we considered: Pulse-Width Modulation, RGB LED and both brushed and stepper motor controller.

1) Pulse-Width Modulation: Pulse-Width Modulation (PWM) is probably the most popular and simple technique to modulate a physical quantity. PWM consists in the generation of a rectangular waveform - with both a programmable period and a duty cycle - to modulate the average value of the physical quantity.

We implemented a PWM controller with an 8-bit programmable period and duty cycle in the eFPGA using 7 of the 16 CLBs. We synthesized the same HDL code in an ASIC version and obtained a block of 156 equivalent gates. Since PULPino does not have a PWM peripheral, in order to implement a PWM controller we needed to use two timers to make the timing and the ISRs manage I/Os while the core was in sleep mode to reduce the power consumption.

In order to have an idea of a commercial microcontroller equipped with a PWM peripheral, we measured the power consumption of the ultra-low-power microcontroller STM32L152RE, setting the processor in low-power sleep mode. Initially, we carried out the measurements switching off all the peripherals and setting GPIOs in analog mode, then we activated only timer 3 (TIM3) and GPIO port B. The power consumption change is due to the peripheral used to implement the PWM controller and is scaled in BCD technology following equation (2). The resulting energies per task are: 16.69 pJ for eFPGA, 7576.8 pJ for PULPino, 0.267 pJ for ASIC and 3.185 pJ for STM32 as reported in Table IV.

Obviously, ASIC has the best energy efficiency despite lacking programmability. The absence of a PWM peripheral in PULPino has a negative impact on energy efficiency because the core has to wake up from sleep mode, fetch and execute instructions just to update a few bits of the FSM. Implementing a PWM controller employing an eFPGA rather than PULPino allows one to achieve an energy gain E_{GAIN} of 454. Use of the STM32 peripheral yields good performances in terms of both power consumption and programmability - these kinds of peripheral have many configuration parameters - without achieving the wide reconfigurability of the eFPGA.

2) RGB LED Controller: An RGB LED controller has to generate three different PWM signals for any color component (Red, Green and Blue) [42]. Modulation of a channel duty-cycle produces an alteration in a color component. Moreover, the RGB LED controller is very similar to the previous application, hence the analysis is the same. We designed a controller with an 8-bit programmable period and 3 × 8-bit programmable duty cycles for the three color components. The controller needs 11 CLBs and the energy per task of the eFPGA E_{eFPGA} is 22.76 pJ. The ASIC implementation requires just 260 equivalent gates and its energy per task E_{ASIC} is 0.373 pJ. This controller was not implemented in PULPino because of the lack of required hardware support (e.g. PWM and enhanced timer) resulting in a too inefficient implementation making the processor busy all the time producing a straightforward additional energy waste. On the contrary, the exploitation of available peripherals makes the implementation on STM32 very efficient. In this solution, we used the previous approach, keeping the processor in low-power sleep mode and using TIM4 timer with 3 channels and port B of GPIOs. The energy performance E_{STM32}, computed as for the PWM controller, is 5.282 pJ. The whole application data are summarized in Table IV.

3) Brushed Motor Controller: Brushed motors are very common DC motors and are typically controlled by half or full H-bridge circuits [43]. Semiconductor companies have developed many dedicated ICs for brushed motor control either

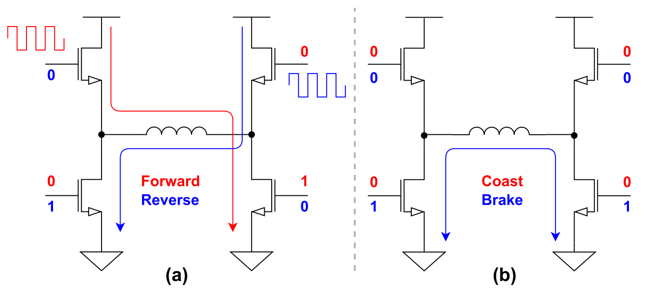


Fig. 8. Brushed controller H-bridge operating modes. (a) Forward - red - and reverse - blue - mode. (b) Coast - red - and brake - blue - mode

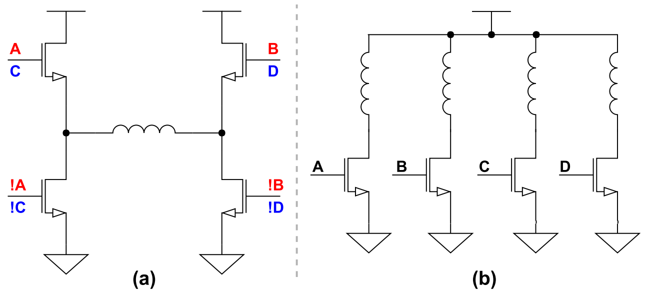


Fig. 9. Driver circuits for stepper motors. Two full H-bridges - respectively red and blue inputs - for bipolar stepper motors (a). Transistor scheme for unipolar stepper motors (b).

with or without power devices. We designed the HDL code of a brushed motor controller for full H-bridge. The controller generates the four driver signals for forward, reverse, coast and brake mode as shown in Fig. 8. For forward and reverse mode - Fig. 8(a) - one needs a PWM signal - 8-bit programmable period and duty cycle - to modulate the motor rotation features. In stop mode, the motor is free to relax in coast mode or is abruptly stopped in brake mode - Fig. 8(b).

The HDL code is synthesized by the eFPGA tool and uses 7 eFPGA CLBs with an energy per task E_{eFPGA} of 18.2 pJ. The PULPino solution uses timers, an interrupt controller and GPIO peripherals to generate both a PWM signal and output signals. To update the FSM states PULPino executes 110 assembly instructions, and the resultant energy per task is 11907.5 pJ. The ASIC implementation has 179 equivalent gates and the energy per task E_{ASIC} is 0.241 pJ. The STM32 implementation uses a timer, interrupt controller and GPIO peripherals and to update the FSM state it executes 79 assembly instructions corresponding to an energy per task E_{STM32} - scaled in our technology - of 10515.24 pJ. Both microcontroller implementations are comparable in terms of architecture - RI5CY and ARM Cortex-M3 respectively 4-stage and 3-stage [44] RISC pipeline in Harvard architecture. Thus, they have similar energy per task as reported in Table IV. The eFPGA energy gain E_{GAIN} is 878.

4) Stepper Motor Controller: Stepper motors are brushless DC electric motors [45]. They have full rotation divided into equal steps. For this reason stepper motors are typically driven in open-loop without a negative feedback. There exist both bipolar and unipolar stepper motors that have the same controller but different electronic power circuits. Fig. 9(a)

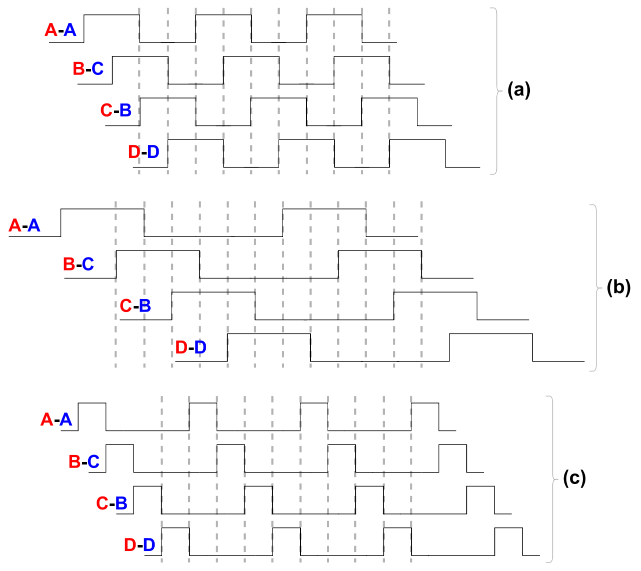


Fig. 10. Signals patterns for unipolar - red - and bipolar - blue - for: full-step (a), half-step (b) and wave-drive control modes.

shows two H-bridge circuits for bipolar stepper motors and each H-bridge has red or blue inlets - where !x means not(x). Figure 9(b) shows a power circuit for unipolar stepper motors. We designed a controller for full-step, half-step and wave-drive control modes and its generated signal patterns are reported in Fig. 10 for unipolar stepper motor - red - and bipolar - blue. When stop input is activated, the controller generates no more signals even though one or two remain active - depending on the control policy. Hence, to avoid motor short circuit current while keeping the torque, programmable PWM signals - 8-bit period and duty cycle - are generated replacing the active phases.

The controller is implemented in the eFPGA using 4 CLBs and by acting on the prescaler it is possible to set the time unit - vertical grey dashed line of Fig. 10 - and the latency of the controller. The eFPGA energy per task E_{eFPGA} is 17.49 pJ. The same HDL code is synthesized in an ASIC solution and the resultant area is 340.26 μm^2 with 125 equivalent gates. The corresponding energy per task E_{ASIC} is 0.419 pJ. For the PULPino solution we used the previous approaches. Timers were used to generate the phase time unit and the processor handled the FSM state. GPIOs and interrupt controller were also used. The assembly instructions to manage FSM are 145 and the energy per task E_{PULP} is 15346.8 pJ. In this case the energy gain E_{GAIN} is 877. The STM32 core executes 111 assembly instructions to manage the controller FSM, also employing a timer, interrupt controller and GPIOs. The corresponding energy per task is 15009 pJ.

B. Bitwise Streaming Applications

In bitwise streaming applications the goal is to make some kinds of operation to compute data. Thus, unlike control applications, the streaming application computational base model is data flow and not a finite state machine. Microprocessors have an intrinsic data flow in their pipeline while eFPGA circuits and ASIC have to be configured as a specific elaboration unit. Hence, a processor in this case exploits its structure better, especially if a necessary operation is mapped in the instruction set architecture. As examples we considered the generation of both an error-detecting code (Cyclic Redundancy Check) and pseudo-random numbers. These sample applications are commonly used in communication protocols, storage devices and cryptography.

1) Cyclic Redundancy Check: Cyclic Redundancy Check (CRC) is an error-detecting code. We considered CRC16 with a programmable polynomial - as an example we used $x^{16} + x^{15} + x^2 + x^0$ polynomial, typically used in both Modbus industrial communication protocols and USB.

The CRC16 block generator is synthesized in the eFPGA using 4 CLBs and at every clock cycle it generates a CRC with an energy of 51.88 pJ.

For the PULPino implementation we used a fast-CRC algorithm based on hash tables [46]. In this case to compute the CRC one needs 8 assembly instructions n_{insn} with an energy per task E_{PULP} of 840 pJ as shown in Table IV. The energy gain E_{GAIN} achieved is 16 which is smaller than that reached in control applications. To obtain the same throughput as the eFPGA, supposing execution of one instruction per clock cycle, PULPino requires a clock frequency f_{PULP} of:

$$f_{PULP} = \frac{f_{eFPGA}}{n_{tick}} \cdot n_{insn} \tag{8}$$

The STM32 μC has quite constant power consumption when the pipeline is working and in this case, to compute CRC - using hash tables - one needs 10 assembly instructions n_{insn}, which corresponds to an energy per task E_{STM32} of 1574 pJ.

The ASIC solution of CRC is no more expensive than other applications in terms of area occupation - 554.3 μm^2 and 201 equivalent gates but it is hungrier in terms of power consumption, its energy per task E_{ASIC} being 1.13 pJ as reported in Table IV. This means that the increase in average power consumption is due to an increase in switching activity. Obviously the ASIC implementation still allows the best energy performances, but if you want to change the CRC polynomial or data-width you have to redesign the whole circuit while for reconfigurable solutions you only need to reprogram the circuit.

2) Pseudo-Random Number Generation: The easiest way to generate pseudo-random numbers is through Linear Feedback Shift Registers (LFSRs). We designed a 16-bit pseudo-random number generator with an LFSR. The generator is implemented in the eFPGA using 15 of the whole 16 CLBs with an energy per task E_{eFPGA} of 88.78 pJ.

The technique used for the PULPino solution is the same as used for the CRC16 and is based on fast-LFSR which uses hash tables executing 42 assembly instructions. The generator mapped on the PULPino has 3721.2 pJ energy per task E_{PULP} and the corresponding eFPGA energy gain E_{GAIN} is 42 which is smaller than control applications. As for the CRC example, to achieve the same throughput as the eFPGA, the PULPino clock frequency has to be defined by equation (8) - much higher than 10 MHz.

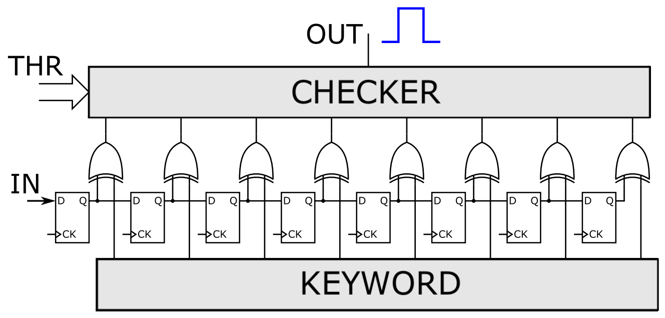


Fig. 11. Wake-up radio correlator architecture.

The STM32 executes 15 assembly instructions to generate a pseudo-random number with an energy per task E_{STM32} of 2170.86 pJ. Due to task complexity, ASIC implementation needs both more area - 1174.4 μm^2 - and more equivalent gates 442 - than previous applications. The power consumption of the ASIC also increases. The greater device area causes an increase in power consumption - P_{ASIC} Table IV - in terms of the dynamic contribution, since power leakage is negligible in this technology. In this case, the overall power consumption is 29.3 μW and only 10.3 nW is due to power leakage - about 1/3000.

C. Ultra-Low-Power Applications

In this type of application the main aim is to have the smallest possible power consumption, which makes software-programmed processors unsuitable due to the intrinsic energy overhead. Hence, for this kind of application ASICs are typically used. Nowadays, every object has to be connected to every other one using as little energy as possible. For this reason, communications systems have to reduce their power consumption since they strongly affect the overall system power consumption. One possible solution is to use Wake-Up Radio (WUR) [47], an always-on circuit capable of recognizing an address/keyword in the received bitstream as well as waking up the system. We designed the 8-bit correlator shown in Figure 11. It receives the serial bitstream from the analog WUR front-end and bitwise compares the bitstream with a predefined keyword - xor gates. The comparator outputs are connected to the combinational logic checker - Fig. 11. The checker block counts how many “1”s are at comparator xor outputs - that means how many received bits are equal to the keyword - and if the resulting number is greater than a defined threshold - THR of Fig. 11 - the controller generates the wake-up signal - blue line of Fig. 11. The correlator was implemented in the ASIC solution targeting a 100 kHz synthesis clock frequency and the estimation was computed at 100 kHz clock frequency as reported in Table IV. This slow clock frequency was chosen to reduce the dynamic power consumption. The HDL code of the correlator was synthesized in standard cells and the resulting area occupation was 367.7 μm^2 with 134 equivalent gates. The energy E_{ASIC} necessary to compare 8-bits is 0.398 pJ. This is the best solution in terms of energy performance and area occupation, even though it is not reconfigurable. Hence, the eFPGA could be a solution if you need to reuse the circuit with a different address/keyword width or correlation technique. The circuit in Fig. 11 synthesized in the eFPGA uses 5 CLBs and the resulting energy to compare 8 bits is 27.24 pJ.

VII. Results and Discussion

A. Energy Efficiency Consideration

All computed energies per task of each implementation - eFPGA, PULPino, ASIC and STM32 - for any application are reported in Fig. 12 and the data presented in Table IV show the computational base model of both eFPGA and microprocessor implementations.

The average power consumption of the eFPGA P_{eFPGA} is closely related to the application and its dynamic behavior. As shown in Table IV, eFPGA power consumption goes from 20.86 μW - for the simplest task PWM - to 110.95 μW - for the larger and more dynamic LFSR application - which corresponds to about a 5$\times$ increase.

On the other hand, the microprocessor has a defined structure, hence its average power consumption is almost constant - for a single-datapath RISC processor - as shown in Table IV where P_{PULP} and P_{STM32} are indeed roughly constant - respectively from 886 μW to 1082.5 μW and 68.123 μW to 70.338 μW. Thus, if the processor's average power consumption is almost constant, the processor energy efficiency depends on how many instructions it has to execute and hence, how the instruction set architecture fits the applications. As reported in Table IV and Fig. 12, the eFPGA energy per task E_{eFPGA} increases with the computational complexity of the application - solid bars of Fig. 12 - from control applications - a simple FSM computational base model - to bitstream applications - data stream. By contrast, PULPino's energy per task E_{PULP} decreases from control to streaming applications, as shown in Table IV and diagonal bars in Fig. 12. For control applications, processor architecture has an instruction overhead due to prologue and epilogue of the routines necessary for computation consistency, as already shown in [13].

The energy per task gap between eFPGA and PULPino reduces - Fig. 12 - in streaming applications where the pipeline structure is better exploited than control applications, where the processor structure is oversized for the tasks. These energy gaps are defined as an energy gain in equation (7) and are reported in Table V - for each application implemented in both eFPGA and PULPino - in order to justify the area overhead due to the eFPGA peripheral being added in the system-on-chip. The computed energy gains are also reported in the chart in Fig. 13 showing that use of the eFPGA for control applications allows one to increase the energy efficiency of the system-on-chip quite markedly - right side of Fig. 13.

The STM32 results - cross bars of Fig. 12 - show how exploitation of a dedicated peripheral for a specific task - going towards ASIC implementation, dotted bars - enhances the system energy efficiency. For example, in the PWM application STM32 uses almost entirely dedicated peripherals and the processor is switched to sleep mode. Hence, STM32 achieves better energy efficiency than eFPGA. On the other hand, brushed and stepper controller need to use the

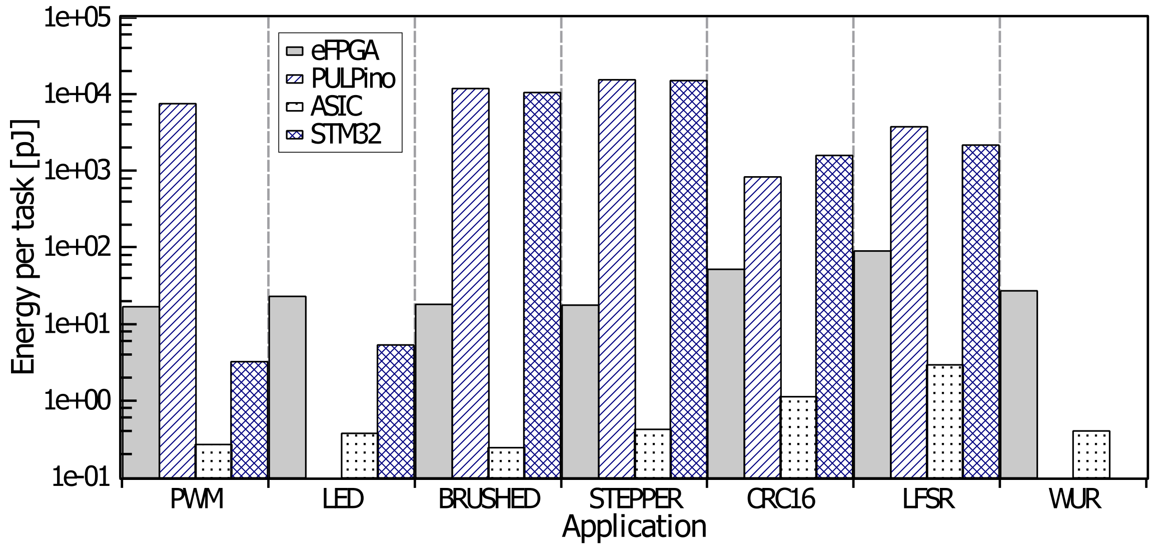


Fig. 12. The energy efficiency derived from equations (3)-(6) - in log scale - and related by number of both CLBs and instructions to carry out the required functionalities.

TABLE V
IMPLEMENTATIONS ENERGY GAIN

	Applications				
	PWM	Brushed	Stepper	CRC16	LFSR
E_{GAIN}	454×	878×	877×	16×	42×

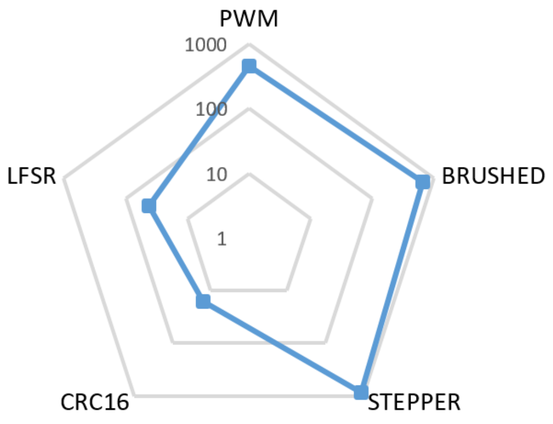


Fig. 13. Energy gain - in log scale - defined by equation (7) for applications implemented in both eFPGA and PULPino.

processor, in addition to dedicated peripherals, to handle the whole FSM, with a straightforward drawback in terms of energy efficiency. However, one may conceivably replace some dedicated peripherals - such as timers, PWM controllers, small pre/post-processing accelerators - with a more reconfigurable peripheral - like the proposed embedded FPGA - extending the usage of peripherals instead of a processor for different kinds of application, which proves more efficient in some cases.

B. Latency Consideration

The eFPGA implementation follows a hardware-paradigm, hence it responds in just 1 clock cycle ($n_{tick} = 1$) just like the ASIC implementation. Microprocessors can achieve the same latency - supposing to execute 1 instruction per cycle which is

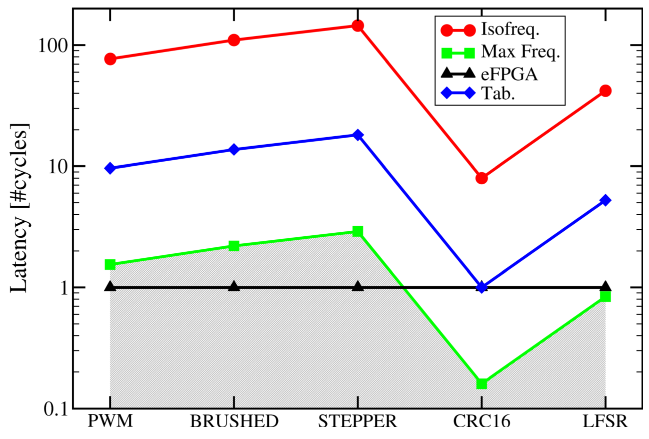


Fig. 14. PULPino and eFPGA latency (in clock cycles) for the applications under analysis.

slightly optimistic - increasing the operating clock frequency. Fig. 14 reports the latency in the number of clock cycles. The eFPGA latency is hence 1 cycle (trace with triangles). Considering the applications in Table IV, if both eFPGA and PULPino have the same clock frequency, PULPino latency is n_{insn} times the eFPGA latency (isofrequency plot with circle symbols). The plot with diamonds reports the case described in Table IV, where PULPino clock frequency is 8 times the eFPGA frequency. As expected, the minimum PULPino latency is achieved when working at the maximum frequency which is the implementation frequency reported in Table III. The corresponding plot is the one with square symbols.

The filled region of Fig. 14 represents the "unreachable latency" zone. This area is out of the PULPino domain since the required clock frequency upscaling is over the implementation frequency.

VIII. CONCLUSION

In this paper we have evaluated the energy benefit of a reconfigurable and fully synthesizable System-on-Chip based on an open-source low-power microcontroller PULPino augmented with a soft-core embedded FPGA. To the best of our knowledge the proposed heterogeneous system which embeds a microcontroller and FPGA is the first SoC suitable for smart power IoT applications. We compared the energy performances of the proposed digital core with ASIC implementations and commercially available STM32 microcontrollers. The analysis focused on the smart power application area such as control-driven, bitwise streaming applications and a wake-up radio correlator for IoT end-nodes. Starting from physical-synthesis results, we analyzed the different power consumption models of microprocessors, eFPGAs and ASICs, to arrive at a comparison between the energy required to execute the same task. The embedded FPGA proves to be an excellent solution in terms of both energy efficiency and latency - despite its non-negligible area overhead - until reaching about three orders of magnitude energy gain. The proposed system-on-chip is very flexible, and thanks to its synthesizability it is portable to different technologies. In addition, combining a microprocessor and eFPGA guarantees wide reconfigurability and efficiency since it is possible to map applications in the best hardware implementation - switching off the processor while using eFPGA and vice versa, or using interaction of both. We have demonstrated how a soft-core eFPGA may efficiently be used for unconventional FPGA tasks such as smart power applications, unlike high-density parallel computing.

References

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

[2] E. Lupon, S. Busquets-Monge, and J. Nicolas-Apruzzese, "FPGA implementation of a PWM for a three-phase DC–AC multilevel active-clamped converter," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1296–1306, May 2014.

[3] M. Ricco, L. Mathe, and R. Teodorescu, "FPGA-based implementation of sorting networks in MMC applications," in *Proc. IEEE 18th Eur. Conf. Power Electron. Appl. (EPE ECCE)*, Sep. 2016, pp. 1–10.

[4] E. Monmasson, L. Idkhajine, and M. W. Naouar, "FPGA-based controllers," *IEEE Ind. Electron. Mag.*, vol. 5, no. 1, pp. 14–26, Mar. 2011.

[5] E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 224–243, May 2011.

[6] STMicroelectronics. (Mar. 2019). *STSPIN Datasheet*. [Online]. Available: http://www.st.com/en/motor-drivers.html

[7] *STLUX Datasheet*, document ID027870, Review 1, STMicroelectronics, May 2015.

[8] B. Murari, F. Bertotti, and G. A. Vignola, *Smart Power ICs: Technologies and Applications* (Springer-Series in Advanced Microlectronics), vol. 6. Berlin, Germany: Springer-Verlag, 2002.

[9] A. Andreini, C. Contiero, and P. Galbiati, "A new integrated silicon gate technology combining bipolar linear, CMOS logic, and DMOS power parts," *IEEE Trans. Electron Devices*, vol. ED-33, no. 12, pp. 2025–2030, Dec. 1986.

[10] M. Sanzaro, P. Gattari, F. Villa, A. Tosi, G. Croce, and F. Zappa, "Single-photon avalanche diodes in a 0.16 μm BCD technology with sharp timing response and red-enhanced sensitivity," *IEEE J. Sel. Topics Quantum Electron.*, vol. 24, no. 2, pp. 1–9, Mar. 2018.

[11] Z. Dong *et al.*, "An integrated transmitter for LED-based visible light communication and positioning system in a 180 nm BCD technology," in *Proc. IEEE Bipolar/BiCMOS Circuits Technol. Meeting (BCTM)*, Sep./Oct. 2014, pp. 84–87.

[12] M. Rose and H. J. Bergveld, "Integration trends in monolithic power ICs: Application and technology challenges," *IEEE J. Solid-State Circuits*, vol. 51, no. 9, pp. 1965–1974, Sep. 2016.

[13] F. Renzini, D. Rossi, E. F. Scarselli, C. Mucci, and R. Canegallo, "A fully programmable eFPGA-augmented SoC for smart-power applications," in *Proc. IEEE 25th Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2018, pp. 241–244.

[14] *STM32L151xE STM32L152xE Datasheet*, document ID025433, Review 9, STMicroelectronics, Aug. 2017.

[15] PULP-Platform. (Jan. 2019). *PULPino Datasheet*. [Online]. Available: https://www.pulp-platform.org/implementation.html

[16] Xilinx. (Mar. 2019). *Xilinx System-on-Chips*. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc.html

[17] Intel. (Mar. 2019). *Intel System-on-Chips*. [Online]. Available: https://www.intel.com/content/www/us/en/products/programmable/soc.html

[18] QuickLogic. (Mar. 2019). *Embedded FPGA*. [Online]. Available: https://www.quicklogic.com/

[19] Lattice Semiconductor. (Mar. 2019). *Lattice Products*. [Online]. Available: http://www.latticesemi.com/

[20] Microsemi. (Mar. 2019). *Microsemi Products*. [Online]. Available: https://www.microsemi.com/

[21] Menta. (Mar. 2019). *Embedded Programmable Logic*. [Online]. Available: http://www.menta-efpga.com/

[22] Flex Logix. (Mar. 2019). *Embedded FPGA Basics*. [Online]. Available: http://www.flex-logix.com/fpga-tutorial

[23] Achronix. (Mar. 2019). *The Speedcore Embedded FPGA*. [Online]. Available: https://www.achronix.com/

[24] J. H. Kim and J. H. Anderson, "Synthesizable standard cell FPGA fabrics targetable by the verilog-to-routing CAD flow," *Trans. Reconfigurable Technol. Syst.*, vol. 10, no. 2, 2017, Art. no. 11.

[25] M. Cuppini, C. Mucci, and E. F. Scarselli, "Soft-core embedded-FPGA based on multistage switching networks: A quantitative analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 12, pp. 3043–3052, Dec. 2015.

[26] M. Borgatti, F. Lertora, B. Forêt, and L. Calí, "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.

[27] 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.

[28] D. Rossi, F. Campi, S. Spolzino, S. Pucillo, and R. Guerrieri, "A heterogeneous digital signal processor for dynamically reconfigurable computing," *IEEE J. Solid-State Circuits*, vol. 45, no. 8, pp. 1615–1626, Aug. 2010.

[29] *STM32TM32-Bit MCU Family*, document BRSTM320218, STMicroelectronics, Feb. 2018.

[30] *MSP430FR604x, MSP430FR504x Ultrasonic Sensing MSP430TMmicrocontrollers for Gas and Water Flow Metering Applications Datasheet, SLASEF5*, Texas Instrum., Dallas, TX, USA, Jan. 2019.

[31] *STSPIN32F0 Advanced BLDC Controller With Embedded STM32 MCU*, document ID029806 Review 2, STMicroelectronics, Mar. 2017.

[32] *nRF52 Series SoC, nRF52 Series PB Version 3.0*, Nordic Semicond., Trondheim, Norway, Mar. 2019.

[33] *SAC57D54H Data Sheet: Technical Data, NXP Semiconductors*, document SAC57D54H, Review 7, May 2017.

[34] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[35] S. J. Wilton, C. H. Ho, B. Quinton, P. H. W. Leong, and W. Luk, "A synthesizable datapath-oriented embedded FPGA fabric for silicon debug applications," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 1, no. 1, 2008, Art. no. 7.

[36] F. Renzini, M. Cuppini, C. Mucci, E. F. Scarselli, and R. Canegallo, "Quantitative analysis of multistage switching networks for embedded programmable devices," *Electronics*, vol. 8, no. 3, p. 272, 2019.

[37] D. Rossi *et al.*, "Application space exploration of a heterogeneous run-time configurable digital signal processor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 2, pp. 193–205, Feb. 2013.

[38] M. Cuppini, "Methodologies for synthesizable programmable devices based on multi-stage switching networks," Ph.D. dissertation, ARCES, Univ. Bologna, Bologna, Italy, May 2015. [Online]. Available: http://amsdottorato.unibo.it/7013/1/Cuppini_Matteo_Tesi.pdf

[39] Synopsys. (Mar. 2019). *Design Compiler Tool in Graphical Mode*. [Online]. Available: https://www.synopsys.com/

[40] *STM32L1xx Ultralow Power Features Overview Datasheet (AN3193)*, document ID17369, Review 2, STMicroelectronics, Sep. 2013.

[41] G. Baccarani, M. R. Wordeman, and R. H. Dennard, "Generalized scaling theory and its application to a 1/4 micrometer MOSFET design," *IEEE Trans. Electron Devices*, vol. ED-31, no. 4, pp. 452–462, Apr. 1984.

[42] *iCE40 Led Driver Usage Guide, Technical Note TN1288*, Lattice Semicond., Hillsboro, OR, USA, Jun. 2016.

[43] R. Condit, "Brushed DC motor fundamentals," Microchip, Chandler, AZ, USA, Tech. Rep. AN905, Jan. 2004.

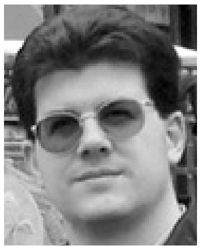
[44] S. Sadasivan, "An introduction to the ARM cortex-M3 processor," ARM, White Paper, Oct. 2006.

[45] C.-H. Liu, J.-S. Ji, and X.-P. Chen, "Control module for stepper motor based on FPGA," in *Proc. IEEE Int. Conf. E-Product E-Service E-Entertainment*, Nov. 2010, pp. 1–3.

[46] C. Mucci *et al.*, "Implementation of parallel LFSR-based applications on an adaptive DSP featuring a pipelined configurable gate array," in *Proc. ACM Conf. Design, Automat. Test Europ*, 2008, pp. 1444–1449.

[47] A. Elgani *et al.*, "Nanowatt wake-up radios: Discrete-components and integrated architectures," in *Proc. IEEE 25th Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2018, pp. 793–796.

Francesco Renzini received the M.Sc. degree in electronic engineering from the University of Bologna, Italy, in 2015, where he is currently pursuing the Ph.D. degree with ARCES. He worked for almost a year as an Embedded Systems Designer for industrial applications with the Research and Development Department of a company. His current research interests regard reconfigurable SoCs for smart power applications.



Claudio Mucci received the degree in electronic engineering and the Ph.D. degree from the University of Bologna, Italy, in 2003 and 2007, respectively. Since 2003, he has been with ARCES, and during that period, he has also been a Consultant in reconfigurable computing with STMicroelectronics, Agrate Brianza, Italy. In 2009, he joined STMicroelectronics Technology R&D. His main research interests include embedded configurable platforms, digital signal processing, application development, and related methodologies. He is coauthor of about 40 publications in international conferences and journals.

Davide Rossi received the Ph.D. degree from the University of Bologna, Italy, in 2012. Since 2015, he has been a Post-Doctoral Researcher with the "Guglielmo Marconi" Department of Electrical, Electronic and Information Engineering, University of Bologna, where he is currently an Assistant Professor. His research interests include energy-efficient digital architectures in the domain of heterogeneous and reconfigurable multi- and many-core systems on a chip. In this field, he has published more than 80 papers in international peer-reviewed conferences and journals.

Eleonora Franchi Scarselli (M'98) received the M.S. degree in electrical engineering and the Ph.D. degree in electrical engineering and computer science from the University of Bologna, Bologna, Italy, in 1992. From 1994 to 2004, she was a Research Assistant with the Faculty of Engineering, University of Bologna, where she has been an Associate Professor since 2005 and currently teaching the design of digital integrated circuits. She is a member of the Scientific Committee of the joint STMicroelectronics-ARCES Laboratory. Her main research activities are in the field of embedded reconfigurable devices and architecture and circuits for low-power sensing and actuating IoT nodes.

Roberto Canegallo received the degree in electrical engineering from the University of Pavia and the Ph.D. degree from the University of Bologna, Italy. Since 1992, he has been with STMicroelectronics, where he worked in Technology R&D on multilevel Flash memories for five years. In 1998, he was with the Research Laboratory jointly managed by STMicroelectronics and Bologna University as a Project Manager for innovative design of IC solutions. He is currently a Senior Member of the Technical Staff at STMicroelectronics and a Program Manager at the R&D smart power technology joint STMicroelectronics-ARCES University of Bologna Laboratory. His main activities are in the field of sensors, power electronics, and ultra-low-power solutions for Internet-of-Things application in smart homes. He has authored many scientific papers and holds European and U.S. patents.