A Fully Programmable eFPGA-Augmented SoC for Smart-Power Applications

A Fully Programmable eFPGA-Augmented SoC for Smart-Power Applications

F. Renzini, D. Rossi, E. Franchi Scarselli
ARCES – DEI, University of Bologna
Bologna 40126, Italy
{francesco.renzini, davide.rossi, eleonora.franchi}@unibo.it

C. Mucci, R. Canegallo
STMicroelectronics
Agrate Brianza 20864, Italy
{claudio.mucci, roberto.canegallo}@st.com

***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, featuring digital elaboration with small computational load, at a lower power consumption and higher responsiveness compared to processor-based implementation. Added value of the proposed system is that the whole digital system is synthesizable, since also the eFPGA is based on a soft-core approach. In the paper we discuss the application domain and present the results of integrating an eFPGA with a computational capability of ≈1K equivalent gates in the STMicroelectronics 0.13 µm Bipolar, CMOS, DMOS (BCD) Smart-Power technology featuring only four metal layers. As expected, eFPGA integration in the SoCs introduces a significant area-overhead (about 20÷25%), but has a straightforward benefit in terms of energy consumption reduction compared to processor-based implementations. On average, based on our analysis, the energy gain achievable in this scenario can be quantified in a couple of orders of magnitude.**

Keywords—Smart Power; Microcontroller; Embedded FPGA;

I. INTRODUCTION

In the Internet of Things (IoT) era, the widespread interaction of the end-nodes with the real-world (cyber-physical interaction) requires systems capable of sensing the environment, elaborating the acquired data, transmitting compressed information and - last but not least - providing a feedback to physical objects. In this work we focus on Smart-Power applications (e.g. motion- and lighting-control [1] [2]), which are mostly based on a control-driven paradigm which leverages simple controllers - e.g. finite-state machines (FSM) - with few inlets/outlets and states.

Traditional Smart-Power SoCs feature analog circuits (for sensing), hardwired FSMs (for control) and high-power circuits (for driving), leveraging the capabilities of Smart-Power technologies such as BCD which integrates bipolar (for analog), CMOS (for digital) and DMOS (for high-voltage) transistors. The limited flexibility of the digital control in traditional systems is mainly caused by the slower scaling of technology nodes implementing BCD with respect to CMOS technologies, driven by the need to include high-voltage devices on the same manufacturing process. These scaled processes allow to integrate more complex digital circuits in Smart-Power SoCs, such as small (ARM Cortex-M class) processors. Such integration between software programmable processors and power electronics is enabling new generations of Smart-Power IoT applications.

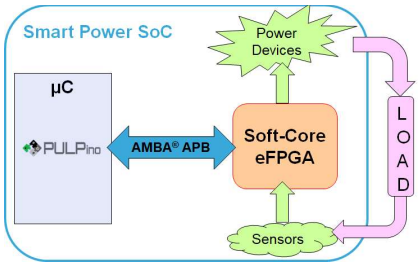

Fig. 1. Reconfigurable Smart-Power SoC.

On the application side, while most traditional ASIC approach does not allow for reuse of the same system for multiple applications, typical programmable digital controllers usually follow two approaches, based on microprocessors and programmable logic devices (PLDs). Pioneeristic works [3] and [4] show the silicon realization of hybrid systems, with the processor coupled to an eFPGA as data computer for either high-density elaborations, coprocessing or configurable I/Os for customizable external peripherals. In this context of multipurposeness, a relevant evolution of eFPGA architecture has been presented in [5], through the introduction of a new interconnect network, based on a regular multiplexer-based structure, which shows properties looking toward full softcore approach, hence full synthesizability. On the other hand, all those previous works were realized targeting advanced CMOS technologies and high performance eFPGAs were implemented as custom-designed hard-macros technique for area/performance optimization.

The introduction of reconfigurable SoCs in the Smart-Power arena is a new challenge for the technologies driving this area, due to the peculiarity of both system and technology. In this context, we present an energy-aware analysis of a Smart-Power SoC integrating a soft-core eFPGA architecture [6] tailored to small-computational tasks within the open-source PULPino microcontroller system [7] on STMicroelectronics 0.13 µm BCD technology.

II. SYSTEM-LEVEL ARCHITECTURE

The digital system is composed of the open-source ultralow-power microcontroller PULPino [7] coupled with our softcore eFPGA [6]. Interfacing is done through AMBA Advanced Peripheral Bus (APB) as shown in Fig. 1. Therefore, the eFPGA

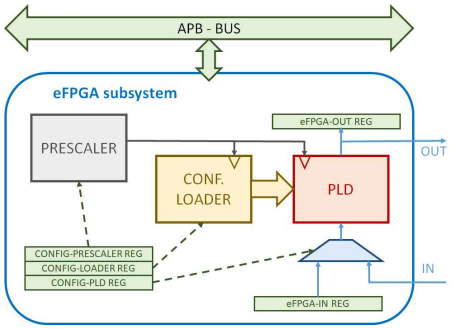

Fig. 2. The eFPGA subsystem interfaced through APB bus.

can be configured by the processor to implement small controllers as Pulse-Width Modulator (PWM) and pure Finite-State Machines (FSMs). The whole system is controlled by the processor, that can be switched in sleep-mode to reduce the dynamic power consumption when computation is not required.

PULPino processor core is based on an implementation of the RISC-V instruction set architecture optimized for energy-efficient digital signal processing [9]. PULPino is also equipped with the traditional peripherals e.g. serial communication interfaces, timers, interrupt controller. The system described in this paper is implemented with 32-bit 4K-Words static RAM for both instruction and data memory.

Soft-core embedded-FPGA [6] we adopted is a fully-synthesizable and reconfigurable IP. Fig. 2 represents the eFPGA subsystem, composed of a frequency divider (prescaler), a configuration loader, the PLD core and the interface toward APB, including the configuration registers which allow the processor to handle the subsystem.

Target technology is BCD that is typically based on metal stacks with few levels, which challenge digital design - especially for programmable-logic - in terms of routing congestion. The eFPGA soft-core approach based on standard-cells is advantageous to provide easier floorplan management. In fact, the main area occupation is due to high-power devices and analog sections, then digital section can fully adapt among other parts, contrary to eFPGA hard-macros approach.

Since the target is control-driven application and not high-density computation, the computational capability required has been estimated in ≈1K equivalent-gates, provided by 16 CLBs and 64 primary inlets/outlets. Each CLB has 12 I/Os and 3 Basic Logic Elements (BLEs), which can be used as either 2×LUT 4:2, 2×LUT 5:1 or 1×LUT 6:1. Connectivity of the device is founded on a Multi-Stage Switching Network (MSSN) with butterfly-oriented topology, as shown in [6], which guarantees synthesizability and non-blocking routing properties. As shown in Fig. 2, the inputs and outputs of the eFPGA are directly connected to primary I/Os and to two 64-bit registers (with 8-bit banks) accessible through APB bus, to mix the activity of the processor with that one of eFPGA. The prescaler, fully configured by the processor, is used to divide the system-clock frequency by a factor $n_{div} = 1 \div 2^{15}$ depending on application critical-path and end-user reactivity needs, as well as to sleep the eFPGA subsystem when it is not required. The configuration loader is in charge to load the bitstream into the eFPGA latches,

TABLE I. SYNTHESIS RESULTS

	Min Area	***Max Speed***
Freq.	10 MHz	80 MHz
System Area	3.63 mm^2	3.9 mm^2
eFPGA Area	0.77 mm^2	0.98 mm^2
eFPGA Area %	21 %	25 %

TABLE II. ESTIMATED POWER CONSUMPTION

Power	***Min Area***	***Max Speed***
Core	2.83 mW (49.3%)	22.3 mW (49.3%)
Pheripherals	2.51 mW (43.7%)	20.9 mW (46.3%)
eFPGA (n_{div}=8)	102 µW (1.78%)	912 µW (2%)
Other logic	397 µW (7%)	1.961 mW (4.4%)
Tot.	5.73 mW	45.2 mW

which replace traditional custom SRAM cells. The configuration structure is disabled during the normal operation of the eFPGA, although the processor can whenever program the PLD, PLD I/Os and prescaler thanks to APB-interfaced registers.

III. SMART POWER APPLICATIONS DOMAIN

The "smart" keyword is typically applied when programmability is added to control methodologies [8], implemented through finite-state machines that generate signal patterns for the high/medium power drivers. As an example, FSM-based applications in motion control domain are brushed and stepper motor controllers, digital PWM for power management and lighting control. For such a class of applications energy efficiency is a more and more critical issue. A simple example is a controller for both unipolar and bipolar stepper motors with full-step, half-step and wave-drive control modes. The corresponding computational model is a FSM with ≈10 states (thus implementable with 4 flip-flops) with 5 inputs (clock, enable, direction and 2 mode selectors) and 4 outputs (for phases generation). This example will be used in the following of the paper to show the different energy budget required for the various implementation approaches under analysis. The implementation of Smart Power applications on microprocessor systems is generally based on interrupt policies, because this class of applications is strongly event-driven. For the previous example of the stepper controller, the interrupt service routine of PULPino needs to update the state through a finite-state machine requiring about 46 assembly instructions (RISC-V ISA):

- 15 instructions are used for both prologue and epilogue of the procedure, including instructions required to manage the stack pointer, to avoid interrupt nesting and to save relevant registers used by the interrupt handler;
- 4 instructions are required for the GPIO management;
- remaining 27 instructions are for state control of FSM, performed through 8 branch/jump, 4 arithmetical operations and 15 load/store for data management.

As a general rule, interrupt-based event-driven applications always require the overhead of prologue/epilogue to preserve the consistency of the computation and this payload depends on the complexity of the operation to be executed (e.g. the number of used registers and the need to be saved/restored) whereas the

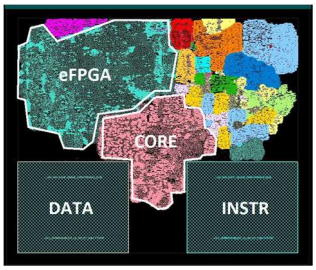

Fig. 3. Physical synthesis floorplan.

same FSM can be directly mapped into the eFPGA. The soft-core approach guarantees more versatility allowing the end- user to define the device based on the application constraints.

IV. Implementation Results

The whole digital section of the SoC has been synthesized with Synopsys Design-Compiler Graphical in 0.13 µm BCD technology of STMicroelectronics, optimizing either area occupation or speed, and the synthesis results are reported in Table I. The frequency target for minimum area has been relaxed down to 10 MHz, while the maximum frequency (max-speed) target is 80 MHz. In both the cases the critical path of the system remains inside the microprocessor, as expected and as required to avoid extra speed-tax due to reconfigurability. For that, the two physical-synthesis show similar area occupations - 3.63 and 3.9 mm^2 respectively - as well as it happen focusing on the eFPGA-subsystem. The area overhead due to the reconfigurable IP is not negligible and it is approximately 21÷25% of the overall SoC area. Fig. 3 shows a synthesis floorplan where is possible to see the two different SRAM macros (bottom-side big rectangles), soft-core eFPGA, RISC-core and other control logics and peripherals. It is clear how the soft-core approach favors area/floorplan optimization contrary to a hard-macros approach, especially when the free space is small and needs to be reshaped depending on top-level Smart-Power IC accommodation.

Table II summarizes the post-synthesis estimated power consumption of the whole digital system for a control-driven application that uses processor for computing and some peripherals i.e. serial communication, general purpose I/Os and eFPGA. Power estimation has been carried out in typical condition (1.8 V and 25 °C) using Synopsys PrimeTime-PX and parasitic back-annotation coming from physical synthesis. The average dynamic power is estimated annotating simulation-based switching activity. The estimated power data shows that eFPGA has a negligible impact - 1.78% for min area and 2% for max speed - on the overall power consumption for $n_{div} = 8$. It can be observed as for this technology the leakage is negligible compared to dynamic power, because the power consumption is about proportional to the clock frequency as shown in Table I and II where there is an ≈8x scaling-factor between max-speed and min-area implementations for operative frequency and dynamic power consumption.

A. Power Consumption Models

RISC processor power consumption is mainly due to memory accesses - read and write - and to pipeline evolution, with an intrinsic relationship to the specific architecture. In general, the power consumption drivers of the processor-based computation are the complexity of the instructions-set architecture (and its bit-width), the number of pipeline stages and the specific memory hierarchy (for data locality optimization). On the contrary, for a given architecture and a given implementation technology, the average power consumption of a single-datapath RISC processor is roughly constant, since the dominant factor is the access to memory for instructions fetching. In our case, for applications with different complexity in Smart-Power sphere, the average power consumption of the processor is 3.15 mW ± 12%, under min-area condition, and to 25.45 mW ± 13% at the max-speed case. As expected, the average values are mostly related to the intrinsic processor architecture, while fluctuations are due to the different applications under analysis and the specific input data. For additional energy saving, it's possible to switch the processor in sleep-mode, thus to obtain a significant power reduction from 2.83 mW to 0.41 mW for min-area and from 22.3 mW to 1.92 mW for max-speed still in typical conditions. In the sleep-mode, processor pipeline is flushed and clock disabled, while all peripherals continue to operate to allow the wake-up when an interrupt or an event occurs. In sleep-mode the switching contribution to power consumption decreases and thus leakage contribution becomes dominant.

The power-consumption model of a programmable-logic device like our eFPGA is similar to that one of a hardwired circuit, although the undeniable overhead due to programmability which act as an upscaling factor (e.g. logical net switching is replaced by the switching of the programmable interconnect structure). As an example, from a logical point of view, the few bits related to a FSM evolution impact the power-consumption on the eFPGA, while on the processor there is the whole activity related to a 32bitwise pipeline evolution.

V. Processor vs. Embedded-FPGA

The power consumption of the processor is hence more or less constant, while for the eFPGA is strictly related to data elaboration. For this reason, we have compared the energy efficiency of both processor and eFPGA architecture to execute some typical Smart-Power elaborations. As an example, the simple motor-drive FSM with few inputs/outputs discussed in Section III has been mapped on 5 CLBs of eFPGA and also implemented in the processor, using interrupt and timer to trigger the routine with the same frequency of the eFPGA and leaving the processor in sleep-mode otherwise. The energy of the eFPGA subsystem is:

$$E_{eFPGA} = \frac{P_{eFPGA}}{f_{eFPGA}} = \frac{P_{eFPGA}}{f_{ck}} \cdot n_{div} \quad (1)$$

where n_{div} is the frequency divider factor (set to 8, in this case). The processor energy for a software-implemented routine is:

$$E_{SW} = E_{per\ ins} \cdot n_{instr} = \frac{P_{CORE}}{f_{ck}} \cdot n_{instr} \quad (2)$$

where $E_{per\ instr}$ is the average energy per instruction, n_{instr} is the number of the instructions of the routine - 46 in this case - and P_{CORE} is the power consumption of the processor that includes RISC-core and memories. We define the energy efficiency as:

$$E_{eff} = \frac{E_{SW}}{E_{eFPGA}} \quad (3)$$

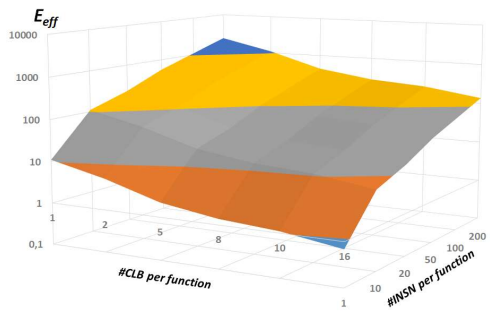

Fig. 4. The eFPGA subsystem interfaced through APB bus.

that compares the necessary energy for both architectures to carry out the same functionality. This model is clearly pessimistic for the eFPGA because, in our estimation, the E_{SW} evaluation doesn't take into account the power consumption of peripherals like timer, interrupt controller and GPIOs. In addition, to simplify the analysis, we don't take into account processor pipeline stalls, assuming to execute one instruction per cycle. For this example, the energy consumption related to processor-based execution of the FSM is around 11 mW/MHz for 46 instructions, while the same task implemented in eFPGA requires just ≈67 µW/MHz. Those numbers take into account the average dynamic power, estimated annotating simulation-based switching activity. The leakage power is a common background and for the technology node we are targeting its contribution in power is roughly one order of magnitude less than dynamic power when the system is fully on.

To provide a general analysis taking care the complexity of both processor- and eFPGA-based implementation, we extrapolated from (1), (2) and (3) an energy-efficiency parametrizing both the number of CLBs and number of assembly instructions required to satisfy a specific functionality. The results are reported in Fig.4, that represents the ratio function between processor energy and eFPGA energy depending on the number of instructions and on the number of CLBs. As visible in Fig. 4, the energy efficiency E_{eff} can be - in log scale - both greater and smaller than 1. When the energy efficiency is lesser than 1, it means that the energy required to eFPGA to carry out the specification is greater than that related to microprocessor. This situation happens, for example, when the task requires arithmetical operations not well mapped on eFPGA (i.e. big number of CLBs) but well matching processor instruction set capability. This case, in Fig. 4, corresponds to ≈16 #CLB and < 2 #insn. On the other hand, when E_{eff} is greater than 1, the energy consumption of the microprocessor is greater than the one of eFPGA: this is quite typical for Smart-Power applications, where the required assignments are usually few-bits FSM-based computation. In those cases, programmable logic devices can implement more efficiently the Smart-Power controller, as a matter of fact, in real applications - where the number of required CLBs is 5÷10 and related processor instructions are 50÷100 - is possible to easily achieve 100÷200 energy gains.

If performances are not relevant at all for the application, it's possible to use other processing cores to further decrease consumption, like the Zero-RISCY [10] [7], a 32-bit processor with just 2-stages pipeline and latch-based register-file which allows to further halve the power consumption during the normal computation. Zero-RISCY has around half-area of RI5CY, thus in sleep-mode also the leakage consumption is halved. In any case, it helps halving the energy efficiency gap, but it doesn't allow to close it, leaving at the best a 50 factor gap.

VI. Conclusion

In this work we have analysed 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. Our analysis focused on the Smart-Power application domain and in particular the control-driven application sector where both the processor and the eFPGA can be used to trigger actuators or to control sensing platform. Starting from physical-synthesis results, we analysed the different power consumption models of both microprocessors and eFPGAs, to derive a comparison between the energy required to execute the same task with the processor or eFPGA. For this application scenario, despite the undeniable area overhead, the embedded-FPGA proves to be an excellent solution in terms of energy efficiency with a couple of order of magnitude gain. Therefore, although challenging for BCD technology, the proposed System-on-Chip can be considered appealing for the Internet-of-Things scenario in Smart-Power applications, where the eFPGA acts more efficiently than the processor for control-driven task, while the main processor can be switched in sleep mode or can manage other computation like communications or data elaborations.

Acknowledgment

This work has been partially funded by R2POWER300 project that has received funding from ECSEL JU Joint Undertaking Call 2014-2, grant agreement n◦ 653933.

References

[1] STLUX385A Datasheet, http://www.st.com/en/power-management/stlux385a.html

[2] STSPIN Datasheet, http://www.st.com/en/motor-drivers.html

[3] M. Borgatti et al., "A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable I/O," IEEE Journal of Solid-State Circuits, 2003, 38.3: 521-529.

[4] A. Lodi et al., "XiSystem: a XiRisc-based SoC with reconfigurable IO module," IEEE Journal of Solid-State Circuits, 2006, 41.1: 85-96.

[5] F.L. Yuan et al., "A multi-granularity FPGA with hierarchical interconnects for efficient and flexible mobile computing," IEEE Journal of SolidState Circuits, 2015, 50.1: 137-149.

[6] M. Cuppini et al., "Soft-core Embedded-FPGA Based on Multistage Switching Networks: A Quantitative Analysis," IEEE Transactions on VLSI Systems, 2015, 23.12: 3043-3052.

[7] PULP-Platform http://www.pulp-platform.org/documentation/.

[8] M. Cuppini et al., "Soft-core eFPGA for Smart Power applications," International Symposium on System-on-Chip (SoC), 2014, 1-4.

[9] M. Gautschi et al., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," IEEE Transactions on VLSI Systems, vol. 25, no. 10, pp. 2700-2713, Oct. 2017.

[10] P. D. Schiavone et al., "Slow and Steady Wins the Race? A Comparison of Ultra-Low-Power RISC-V Cores for Internet-of-Things Applications," PATMOS, 2017.