HEAL-WEAR: an Ultra-Low Power Heterogeneous System for Bio-Signal Analysis

Loris Duch, *Member, IEEE,* Soumya Basu, Rubén Braojos, Giovanni Ansaloni, *Member, IEEE,* Laura Pozzi, *Member, IEEE,* David Atienza, *Fellow, IEEE*

***Abstract*—Personalized healthcare devices enable low-cost, unobtrusive and long-term acquisition of clinically-relevant bio-signals. These appliances, termed Wireless Body Sensor Nodes (WBSNs), are fostering a revolution in health monitoring for patients affected by chronic ailments. Nowadays, WBSNs often embed complex digital processing routines, which must be performed within an extremely tight energy budget. Addressing this challenge, in this paper we introduce a novel computing architecture devoted to the ultra-low power analysis of bio-signals. Its heterogeneous structure comprises multiple processors interfaced with a shared acceleration resource, implemented as a Coarse-Grained Reconfigurable Array (CGRA). The CGRA mesh effectively supports the execution of the intensive loops that characterize bio-signal analysis applications, while requiring a low reconfiguration overhead. Moreover, both the processors and the reconfigurable fabric feature Single-Instruction / Multiple-Data (SIMD) execution modes, which increase efficiency when multiple data streams are concurrently processed. The run-time behavior on the system is orchestrated by a light-weight hardware mechanism, which concurrently synchronizes processors for SIMD execution and regulates access to the reconfigurable accelerator. By jointly leveraging run-time reconfiguration and SIMD execution, the illustrated heterogeneous system achieves, when executing complex bio-signal analysis applications, speed-ups of up to 11.3x on the considered kernels and up to 37.2% overall energy savings, with respect to an ultra-low power multi-core platform which does not feature CGRA acceleration.**

***Index Terms*—Ultra-low power architectures, Coarse-Grained Reconfigurable Arrays, Wireless Body Sensor Nodes, Bio-Medical Signal Processing.**

I. Introduction

ACCORDING to recent studies [1], more than 50 billion devices will be connected to the internet by 2020. A large portion of this amount will be constituted by autonomous devices, the so-called Internet of Things (IoT). The IoT revolution will enable major innovations in the interactions between humans and machines. Among those breakthroughs, it will have an important impact on healthcare provision, greatly lowering the cost for the long-term monitoring of chronic diseases [2], as well as improving the lifestyle of affected patients. This scenario is nowadays especially relevant, since the ageing of the world population and the prevalence of unhealthy habits have turned chronic cardiovascular diseases into the leading cause of death worldwide, even ahead of infectious agents [3]. In this context, wearable IoT devices, also known as Wireless Body Sensor Nodes (WBSNs), have emerged as a cost-effective unobtrusive solution to perform continuous monitoring of clinically-relevant data, outside of a hospital environment and with little supervision from the medical staff [4]. WBSNs are able to autonomously acquire bio-signals with various modalities, such as blood oxygenation (SpO_2), electromyograms (EMGs), etc. Among them, the most common one, on which we focus in this paper, is the electrocardiogram (ECG) [5], that measures the electrical activity of the heart.

A key requirement for WBSN is that they must operate for extended periods of time while relying on small batteries, thus requiring a high energy efficiency. In this context, the energy bottleneck of most WBSNs, which only perform acquisition and transmission, usually resides in the wireless communication stage, because the slow dynamics of bio-signals allow their acquisition while employing little energy [6].

A striking alternative is embodied by "smart" WBSNs, which perform advanced on-board Digital Signal Processing (DSP) to extract high-level relevant features from acquisitions [7]. In this approach, only features (as opposed to samples) are sent through the energy-hungry wireless link, potentially resulting in large energy gains [8], [9]. Nonetheless, these benefits can only be leveraged by performing the DSP stage itself within a small energy envelope. In fact, thanks to progresses in the design of domain-specific analog-to-digital converters [10], [11] and wireless protocols [12], DSP tends to dominate the energy budget of smart WBSNs [13], so that any increase in its efficiency has a tangible impact at the system level.

With the end of Dennard scaling [14], and further reductions in supply voltages hampered by the unreliability of SRAM memories when operating in a near-threshold regime [15], nowadays the most promising opportunities for increasing the energy efficiency of digital processing platforms resides at the architectural level, requiring a careful and domain-specific optimization. In this context, low-power multi-core architectures [15]–[18] have been proposed to leverage the intrinsic code parallelism of bio-signal DSP applications [19]–[21].

Herein we introduce the HEterogeneous and reconfigurable ultrA-Low power architecture for WEARable Body Sensor Nodes (HEAL-WEAR), a domain-specific digital processing platform that operates at ultra-low power levels, harnessing multiple energy-wise optimization opportunities deriving from the application characteristics of bio-signal analysis DSP.

A high-level block scheme of HEAL-WEAR is presented in

L. Duch, S. Basu, R. Braojos and D. Atienza are with the Embedded Systems Laboratory (ESL), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, e-mail: {loris.duch, soumya.basu, ruben.braojoslopez, david.atienza}@epfl.ch.

G. Ansaloni and L. Pozzi are with the Università della Svizzera Italiana (USI), Lugano, Switzerland, e-mail: {giovanni.ansaloni, laura.pozzi}@usi.ch.

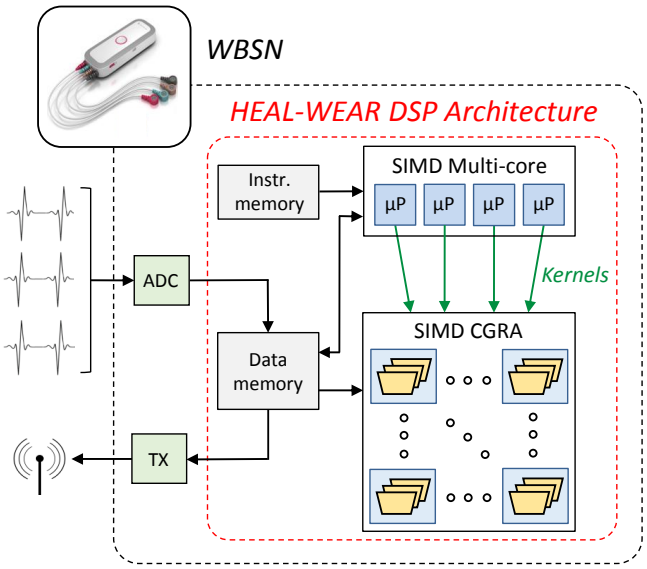

Fig. 1. HEAL-WEAR couples a multi-core processor with a CGRA unit, supporting SIMD execution in both resources.

Figure 1. First, as processing is usually divided in well-defined phases, HEAL-WEAR, similarly to [16], employs multiple homogeneous cores to spread the execution of the control-dominated parts of applications over several processors.

Second, hardware acceleration is provided to efficiently execute the *computational kernels* of the applications, i.e.: compact and intensive code sections, which account for a vast portion of the overall DSP run-time. To maintain a high degree of flexibility, the accelerator is programmable at the operation level, as a Coarse-Grained Reconfigurable Array (CGRA) mesh [22], [23]. In this way, a variety of kernels, possibly unknown at design time, can be supported, avoiding the high area, energy and configuration overhead typical of the fine-grained reconfigurability provided by FPGAs [24]. While private CGRAs could be provided for each processor, this strategy would not be efficient, as it would involve a high degree of resource replication. In HEAL-WEAR, instead, a single CGRA mesh is embedded in the system as a shared resource, which is time- and space-multiplexed among the processors through pipelining and partial reconfiguration.

Third, the system supports Single-Instruction/Multiple-Data (SIMD) execution modes, both during software execution and hardware acceleration. SIMD opportunities are common in bio-signal DSP, which often involves simultaneous processing of different data sources. The HEAL-WEAR architecture leverages SIMD in order to (a) coalesce accesses performed by different processors to the memory subsystem, (b) minimize the number of reconfigurations required to map accelerated loops and (c) streamline the control logic of the CGRA fabric, whose cells embed multiple datapaths. The resulting SIMD-CGRA therefore supports the execution of simultaneous acceleration requests, issued by processors executing in SIMD.

The run-time orchestration of the execution among the resources of such heterogeneous system is not a trivial task, especially when, as in the proposed platform, system management must be achieved with little area, energy and timing overheads. To accomplish it, we introduce a hardware/software mechanism based on a dedicated instruction set extension and a system controller, interfaced with the processors and the CGRA mesh. The controller is able to synchronize the threads on the multiple cores and, at the same time, manage the acceleration requests and their execution on the CGRA.

The energy benefit deriving from the approach embodied in HEAL-WEAR is two-fold. First, by separating the computation- and control-intensive parts of applications, each of them is efficiently mapped on dedicated resources. Second, the speed-up ensuing from hardware acceleration decreases the ratio between active and idle times, which can then be leveraged by supporting aggressive deep-sleep modes.

The main contributions of the paper are the following:

1) We introduce and explore the performance, from an energy-efficiency perspective, of a novel coarse-grained reconfigurable array, optimized for SIMD operations.
2) We detail how the developed SIMD-CGRA can be integrated in an ultra-low power multi-core system devoted to bio-signal analysis.
3) We propose a unified and low-overhead mechanism to jointly support synchronization among cores, acceleration of kernels and power management at the system level.
4) We showcase the energy benefit of the HEAL-WEAR multi-core heterogeneous system when performing complex signal analysis on multi-lead ECG acquisitions.

The rest of the paper is structured as follows: Section II summarizes related works in the field. Section III illustrates the design of the SIMD-CGRA, while Section IV describes how such resource can be efficiently embedded in a multi-core system and shared by multiple processors. Then, Section V describes the developed experimental framework, used to investigate the performance and energy efficiency of HEAL-WEAR. Obtained results are shown and discussed in Section VI. Finally, Section VII summarizes the main conclusions of this work.

II. State-of-the-Art

Embedded bio-signal processing applications have been conceived to monitor a large number of chronic pathologies [25], [26], such as cardiac arrhythmias [27], sleep apneas [28], or Parkinson symptoms [29].

To sustain these complex signal processing workloads at ultra-low power levels, a number of domain-specific digital processors have been proposed. These architectures usually feature a small area footprint in combination with advanced power management schemes, performing dynamic Voltage/Frequency Scaling (VFS) to adapt to different workload conditions at run-time [15], [30], [31]. Aggressive VFS can degrade the performance of computing platforms beyond the real-time requirements of the intended applications. To reclaim a high level of performance, multi-core architectures can be employed [18], [32], exploiting the parallelism intrinsic in bio-signal processing algorithms when multiple signals are concurrently processed. The authors of [33] demonstrated that multi-cores operating at low voltages and frequencies (down to few MHz) outperform single-cores, even for the light workloads that characterize embedded cardiac DSP.

An orthogonal strategy to maximize energy efficiency is the use of dedicated hardware blocks (custom instructions [34] or dedicated accelerators [35]) to efficiently support computationally-intensive segments of applications. While this strategy can result in orders-of-magnitude power reductions, it is also inflexible, as each block can perform a single function. These characteristics are even more pre-eminent when accelerators are shared by multiple cores [36], [37], because requests for accelerated functions must be arbitrated.

In this context, reconfigurable solutions are good candidates to couple the efficiency typical of dedicated hardware with the degree of flexibility required by a wide variety of computational kernels. However, bit-level reconfigurable arrays (such as FPGAs) present huge overheads in terms of area and power consumption, with respect to fixed-function ASICs [24]. CGRAs dramatically reduce these overheads by being programmable only at the operation level, which allows efficient mapping of kernels [22], [38]. Indeed, in [23] the use of CGRAs is advocated based on energy efficiency considerations, while in [39] a coarse-grained array was proposed for the efficient analysis of EEG acquisitions. Scheduling the kernels on CGRA meshes is not straightforward, as operations must be assigned to a spatially distributed RC element, as well as to a temporally defined execution cycle. For the experimental evaluation presented in Section VI, this task was performed manually, mimicking the automated modulo strategy described in [40]. Other CGRA scheduling algorithms include [41], [42], [43] and [44].

Herein, we exploit the parallel nature of coarse-grained reconfiguration by interconnecting a CGRA instance as a shared accelerator in a multi-core system. Our approach has some similarities with the one presented in [45]. Nonetheless, the authors of that work adopt the limiting assumption that the reconfigurable fabric can be accessed by only one core at the time. Conversely, we instead concurrently support requests by different cores, either in a Multiple-Instruction / Multiple-Data (MIMD) or in a SIMD fashion. As opposed to [46], which only considers SIMD at the multi-processor level, our paper investigates the benefits of also supporting SIMD-kernels, adopting a multi-datapath CGRA.

III. Domain-Specific Shared CGRA

Similarly to FPGAs, CGRA architectures are structured as a two-dimensional mesh of Reconfigurable Cells (RCs), tightly interconnected with each other. The structure of the RCs differentiates CGRAs from FPGAs: while the latter can perform any boolean function of the input data, thus providing bit-level flexibility, the functionality of RCs is defined at the operation level, by embedding a dedicated ALU coupled with a small local register file. This arrangement allows CGRAs to efficiently execute Data Flow Graphs (DFGs), extracted from loop-intensive code segments (kernels), in a spatial way [40].

As opposed to fixed-function ASICs, CGRAs can be re-programmed at run-time. Their configuration overhead, as well as the area devoted to the configuration logic, is orders-of-magnitude smaller than that of fine-grained FPGAs, as only the desired ALU operations and the routing of operands must be specified for each cell. Hence, only a short configuration interval is sufficient to provide this information [47]. Multiple operations can be cyclically performed by each cell on the mesh [48], by providing a set of configuration words and activating the proper one at each clock cycle, during execution.

CGRAs are particularly effective in the acceleration of loops. Their structure allows to partially overlap the execution of different iterations, a technique termed modulo scheduling and initially developed for VLIW processors [49]. The speed-up obtained by modulo scheduling loops is determined by the achieved initiation interval, which measures the difference, in clock cycles, between the start of two subsequent loop iterations. The achievable initiation interval for a loop is limited by the amount of RC resources, as well as by the presence of loop-carried dependencies.

A. SIMD-CGRA

The SIMD-CGRA employed in the HEAL-WEAR architecture is composed of 16 RCs, organized in a 4x4 mesh and connected by nearest-neighbour links in a torus configuration. Figure 3 provides a high-level view of the mesh, while Figure 2 details the structure of RCs. A defining feature of the reconfigurable array employed in this work is that, as opposed to state-of-the-art CGRAs [41], our design features multiple datapaths in each cell. In this way, single-instructions/multiple-data execution can be efficiently supported. Datapaths (DPs)

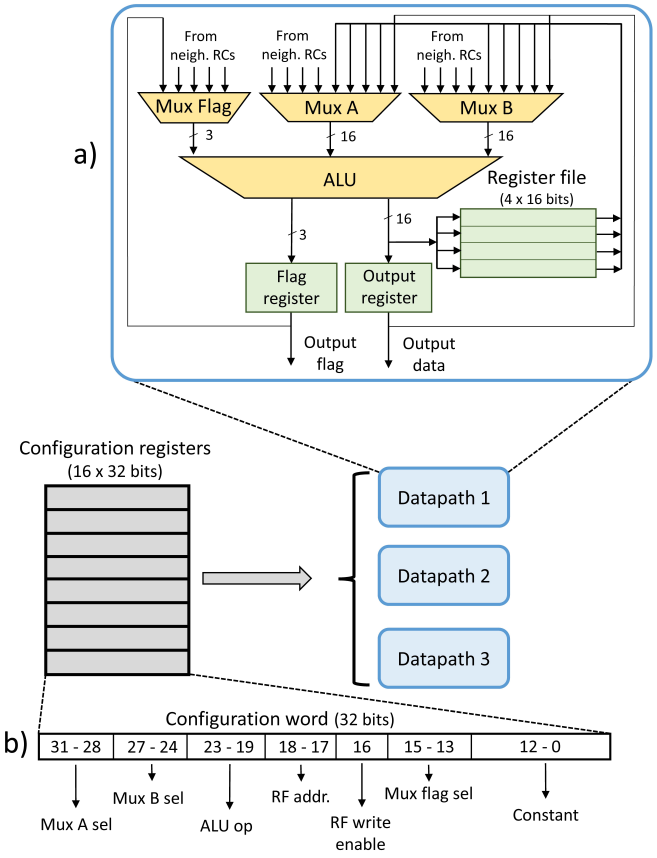

Fig. 2. RC architecture of the SIMD-CGRA, highlighting a) the datapath structure, and b) the format of the configuration words. All datapaths in the RC receive the same configuration word at each clock cycle.

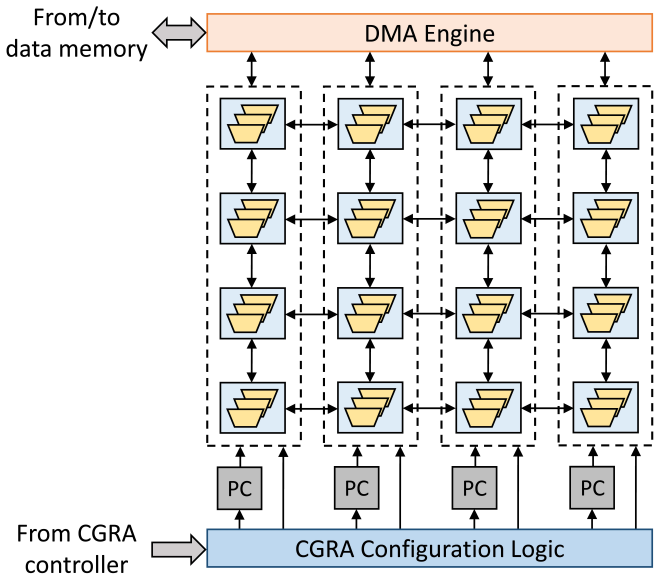

Fig. 3. Block scheme of the SIMD-CGRA.

are composed of an ALU, a local register file, and the multiplexers required to select the input operands (from the register file, or from the output of its own ALU, or from those of neighbouring RCs).

The ALU can execute arithmetic operations (addition, subtraction, multiplication), arithmetic and logic shifts, and bit-wise operations (AND, OR, XOR). In addition, execution branches are supported by if-conversion, i.e., both branches of an "if" statement are executed, and the correct result is then selected based on the outcome of the test condition. To do so, 1-bit flags (zero, negative and overflow) are generated by arithmetic and shift operations. These flags are routed along with data, and used by a dedicated MUX operation. Finally, one ALU per column can perform a square root instruction.

As shown in Figure 2, the DPs belonging to the RCs share the same set of configuration words. Therefore, multiple instances of the same kernel can operate in parallel on different data sets. Instances of the same kernel called by different cores are mapped onto different DPs inside RCs. As kernel instances do not exchange data, no links are present between DPs in a cell, but only among corresponding DPs in neighbouring RCs.

At run-time, the functionality of the DPs of an RC is dictated by its active configuration word, selected among the ones stored in its local configuration registers. Column-wise program counters (named PCs in Figure 3) are in charge of this selection at each clock cycle, so that different kernels can be concurrently mapped on each column of the SIMD-CGRA.

At the periphery of the reconfigurable mesh, a multi-channel DMA block is in charge of providing the interface toward the system data memory. Such transfers operate using data memory ports of the processors that requested the execution of the kernel, therefore they do not require dedicated read and write connections toward the memory subsystem.

B. CGRA Execution Flow

At run-time, each kernel mapped in the CGRA requires a configuration and an execution phase. During configuration, the parameters of the kernel invocation (such as the addresses of input and output data in memory and the number of iterations) are retrieved from the issuing processors and are used to configure the program counters of the employed mesh columns and the required DMA channels. The proper configuration words are then transferred from a dedicated CGRA configuration RAM to the configuration registers of each cell. Kernels may have different sizes, so they may be mapped on multiple columns, up to the entire CGRA mesh. They can also be invoked with a varying SIMD width ranging from one (i.e., no SIMD) to the number of DPs available in each cell. If the SIMD width exceeds the DP width, the kernel invocation is split into multiple copies, mapped on different CGRA regions. If instead the SIMD width of the kernels is less than the number of DPs per cell, unused DPs are power gated to save static and dynamic energy.

Once mapped, the modulo-scheduled [49] kernel is activated and the RCs compute the desired output, which are stored by the DMA engine in the system data memory. To avoid a multi-ported implementation of the CGRA configuration RAM, and a replication of the configuration logic, a single (possibly SIMD) kernel is configured at a time. Execution of different kernels can instead proceed concurrently on separate CGRA columns, effectively employing the available RCs.

IV. Multi-Core System

The SIMD-CGRA mesh described in the previous section is interfaced in the HEAL-WEAR platform with a multi-core system, whose structure and run-time paradigm are detailed in this section. The system leverages the potential of parallel processing at two different levels to maximize its energy efficiency. First, at the thread level, it supports a flexible SIMD execution strategy over multiple cores. Second, at the loop level, it allows for spatial execution of computation intensive kernels (possibly in SIMD) on the coarse-grained reconfigurable mesh.

The system allows for a time- and space-shared utilization of the CGRA module to execute the kernels to be accelerated. Cores are clock-gated when requesting an acceleration, waiting for synchronisation after branches and when no input data is available. CGRA columns (RCs and configuration memory) are power-gated when not in use, as no content has to be retained across subsequent executions of accelerated functions. These features are provided by a unified mechanism supporting fine-grained code synchronization, power management and acceleration of computational hotspots. Its design is detailed in the following parts of this Section.

A. Hardware and software components

The block diagram of the HEAL-WEAR system is depicted on Figure 4. Cores can be individually clock-gated when idle (i.e., when waiting for another core to finish its task or when an acceleration is performed on the CGRA), reducing their dynamic energy requirements.

Processors are interconnected to multi-banked instruction and data memories through combinational crossbars. The data memory space is partitioned in a shared section, devoted to inter-processor communication, and in private sections for

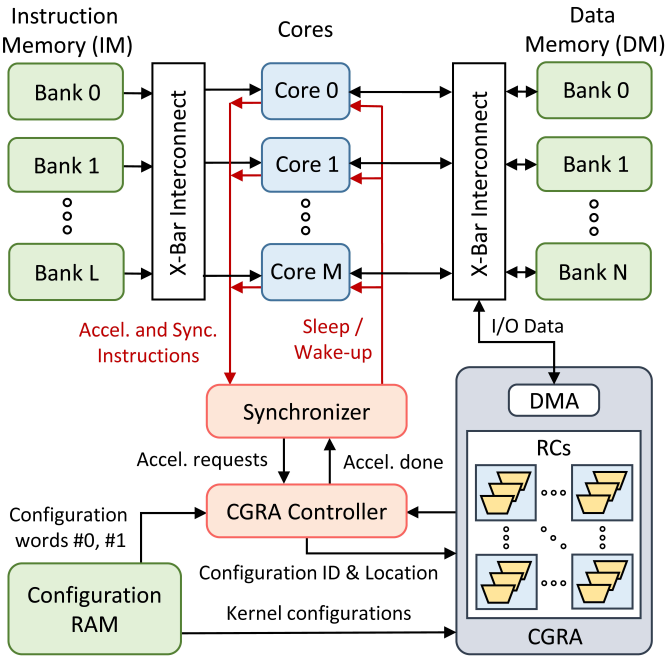

Fig. 4. High-level view of the HEAL-WEAR platform.

each core. Similarly to [50], the crossbars coalesce memory requests when the same data or instruction is read by multiple cores in SIMD mode. This occurrence is frequent in bio-signal applications, since the same signal processing routine is often performed on different input streams, and executed on different processors.

Nonetheless, data-dependent branches can cause the execution flow on cores acting in SIMD to diverge. To recover, at the end of those branches, from run-time divergences, we adopt a solution based on the hardware synchronization mechanism first introduced in [16]. In addition to re-establishing SIMD execution when necessary, the *Synchronizer* of HEAL-WEAR also coordinates the instruction memory accesses (avoiding conflicts) and clock-gates the cores waiting for input data or for the completion of a requested CGRA acceleration.

To accomplish this last task, the Synchronizer is interfaced to a *CGRA Controller*. The role of the controller is to coordinate the acceleration requests from the cores to the CGRA through a request queue. The controller checks that sufficient resources are available at run-time to map an accelerated function. It also arbitrates the access to the reconfigurable accelerator when multiple requests for *different* kernels are received in the same clock cycle. Finally, it is in charge of coalescing requests of the *same* kernels from cores executing in SIMD. In this last case, kernels are concurrently mapped as SIMD-kernels on different DPs of the same reconfigurable cells. In case more SIMD requests are issued than the SIMD width of the CGRA, the ones corresponding to the cores with the lower processor identifier (PID), are executed first.

Storage for the configuration words used to program kernels on the CGRA fabric is provided by a *Configuration RAM*. As detailed in Section III, each word dictates the functionality of a reconfigurable cell (RC) at a given clock cycle during the execution of a kernel iteration.

At the software (instruction set) level, the HEAL-WEAR architecture features the synchronization instructions introduced by the authors of [16]. Calls to these instructions allow proper management of the execution flow of a target application executed on the multiple cores, enforcing energy-

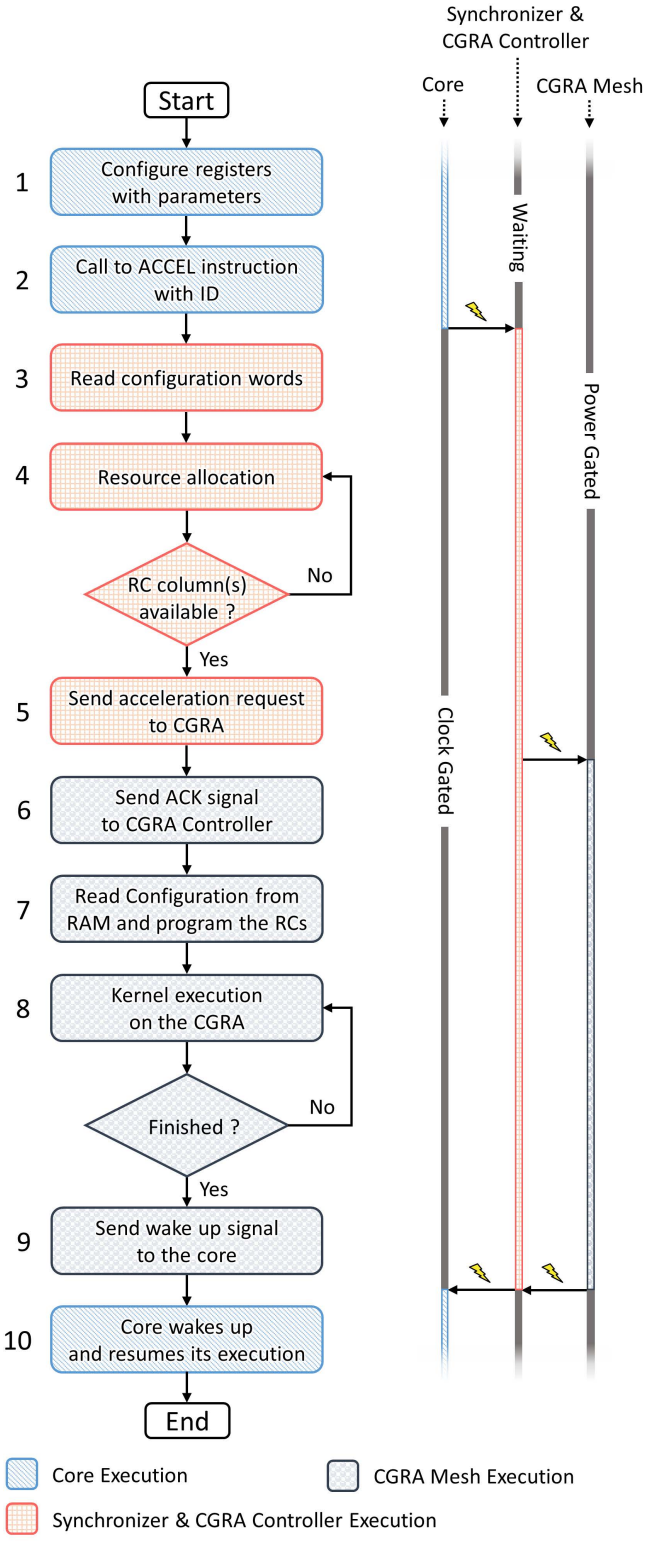

Fig. 5. Acceleration request execution flow diagram.

efficient SIMD execution when possible and managing producer-consumer relationships between threads.

A further instruction set extension supports acceleration requests to the SIMD-CGRA. It is defined as ACCEL *#literal*, where the literal specifies the unique kernel ID that the core wants to execute on the CGRA. The interaction between the hardware CGRA controller and the custom ACCEL instruction is described in the next subsection.

B. Acceleration Request Execution Flow

Acceleration requests by the processors are processed according to the flow-chart presented in Figure 5, whose steps are detailed in the following.

Step 1: Before requesting the execution of a kernel on the CGRA with the ACCEL instruction, each core has to configure a set of private memory-mapped registers, only accessible by the corresponding core and the CGRA. These registers are used to define run-time parameters, which can change from one kernel invocation to another. They specify:

- The address and length of the input data to be processed by the kernel running on the CGRA.
- The address and length of the buffer where results have to be delivered by the kernel executing on the CGRA.
- The number of required loop iterations.
- Possible kernel invariants.

Step 2: After setting the value of these registers, the ACCEL instruction with the correct configuration ID is called by the core. Consequently, an acceleration request signal is raised from the corresponding core and forwarded with the configuration ID to the CGRA Controller through the Synchronizer (cf. Figure 4). In the meantime, the core is clock gated and the acceleration request received by the CGRA Controller is stored into a request queue.

Step 3: For each request popped from the queue, the CGRA Controller reads two configuration words (#0 and #1) from the Configuration RAM. The first word (#0) is used to address and specify the length of the kernel configuration. The second one (#1) specifies the resources (RCs and CGRA columns) required to accelerate the desired kernel. These two words are temporarily stored in local registers until *Step 5*.

Step 4: Before forwarding the request signal to the CGRA mesh, the CGRA Controller searches for a number of free RCs column(s) equal to the ones required by the desired acceleration. Based on a local representation of the current CGRA execution state, this task is performed in three steps:

a) The CGRA Controller checks if this request was already mapped on the CGRA. To do so, the controller considers the required amount of RC columns forming a kernel and tries to find contiguous available RC columns already configured with the requested configuration ID.
b) If this search step fails, the CGRA Controller tries to find the first set of contiguous unused RC columns.
c) If all the RC columns have been used at least once, the CGRA Controller tries to find the first set of contiguous RC columns not in use (i.e. not currently used by a kernel on the CGRA). If insufficient resources are available, the execution is stalled and waits until the release of some RC columns by other kernels.

When available RC columns have been identified, a column index (i.e. configuration location) is determined, specifying the location of the first column of the kernel mapping.

Step 5: When the CGRA Controller finds suitable RC columns to execute the kernel, it sends an acceleration request signal to the CGRA, along with the acceleration ID, the retrieved location and the previously read configuration words.

Steps 6 and 7: The CGRA acknowledges the request and, if necessary, fetches the remaining configuration words corresponding to the accepted acceleration from the Configuration RAM to program the RCs of the used columns.

Step 8: When the kernel is ready, the execution starts. The outputs computed by the kernel are written in data memory, so that they can be used by the requesting core(s) after the kernel completion.

Step 9 and 10: Finally, resources in the local state representation of the CGRA are released, and a wake up signal is sent through the CGRA Controller and the Synchronizer, to the cores which have previously requested the acceleration, thus allowing them to continue their execution.

V. Experimental Setup

In this section, we first describe the architectural parameters of HEAL-WEAR and the framework we developed to investigate its performance. Then, we illustrate the baseline systems. Finally, we detailed the employed benchmarks and the accelerated kernels.

A. Hardware Architecture and Simulation Parameters

To evaluate the energy and performance benefits of the HEAL-WEAR platform, we developed a hybrid framework, which comprises both a post-synthesis and a (higher-level) cycle-accurate view of the system. Its block scheme is presented in Figure 6.

The RTL implementation, cycle-accurate model and compiler of the processors were defined using Synopsys ASIP

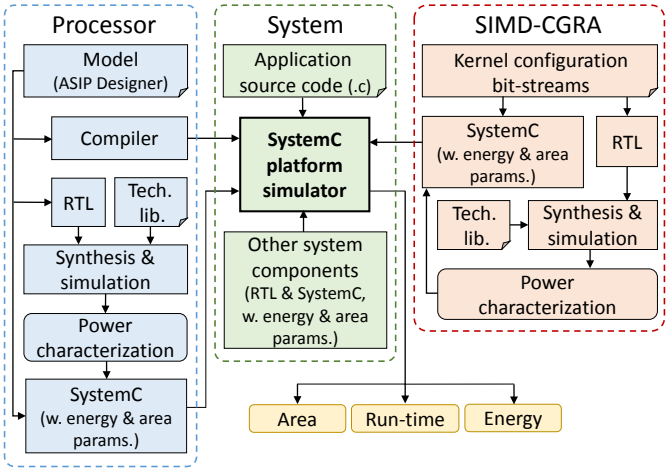

Fig. 6. Block scheme of the experimental framework, comprising RTL and cycle-accurate system views.

Designer [51]. As in [15], each processing core adopts a simple Harvard architecture, featuring a three-stage pipeline. The gate count (12 Kilogates) and the energy-per-cycle of the processors is comparable to that of the low-power ARM Cortex M0 architecture [52].

The SIMD-CGRA accelerator was instead designed from the ground up as an RTL template, allowing the definition of architectural parameters through configuration directives. The mesh comprises 4 rows and 4 columns of RCs, which suffice to map the biggest kernel considered in this study. We considered three different scenarios, having one, two and three DPs per RC, respectively. Its run-time behavior was also modeled in the SystemC simulator. A similar strategy was followed for the other system blocks, such as memories and interconnects.

The complete system includes 8 processors, a data memory of 64 kBytes (32k words of 16 bits) and an instruction memory of 96 kBytes (32k words of 24 bits). The Configuration RAM has a size of 6 kBytes (1.5k words of 32 bits), which is sufficient to store all the configurations of the considered kernels. Each RC has 16 configuration registers, while each DP has a 4-word local register file. The operating frequency is set at 1 MHz for all the benchmarks, except for the computationally intensive 8L-CS benchmark (cf. Section V-C), which operates at 2 MHz, in order to respect the real-time constraint.

Area numbers were directly derived from the synthesized netlist of the system, varying the architectural parameters of the SIMD-CGRA. We employed a 65nm UMC low-leakage technology library, at a supply voltage of 0.9V. To accurately measure the energy consumption, long logic simulations would be required to collect the switching-activities corresponding to the processing of entire ECG windows. Such approach would lead, in turn, to time-consuming simulations at the post-synthesis level which would make the space exploration unfeasible. Instead, post-synthesis simulations were used on a smaller scale, only to perform the energy characterization of the different system blocks. To this end, similarly to the approach in [16], we evaluated (using Synopsys Design Compiler and Mentor Graphics Modelsim) the performance of the multi-processor architecture when executing small synthetic benchmarks, both with and without SIMD execution. In addition, we simulated (again, at the post-synthesis level) the execution of all kernels on the SIMD-CGRA.

Through this analysis we derived detailed energy profiles for each of the system computational components (processors, RCs) when in active and sleep mode, and the power required for memory accesses. The behavior of the system components while in sleep mode depends on whether their state must be retained or not across idle periods. DPs can be safely power-gated when not in use, because values residing in the SIMD-CGRA local register files are not used after a kernel execution. On the other hand, CGRA configuration memories are clock-(but not power-) gated, so that their content can be reused by a further invocation of the same kernel. Processors are also clock-gated when idle, since the content of their internal registers must be preserved at run-time.

The energy profiles, along with the run-time required for the execution of the CGRA kernels, were imported in the cycle accurate (SystemC) simulator, which allows much faster experimental evaluations. Using the simulator, we gathered metrics such as the number of active or inactive clock cycles for each processor and CGRA RC, and data/instruction memory accesses. By combining the information provided at each abstraction level, we were therefore able to gather the performance metrics of the system over long simulated periods of time. At the cycle-accurate level, each benchmark was executed for a total of 10 seconds, corresponding to processing 5000 ECG samples extracted from the T-Wave Alternans Challenge database [53].

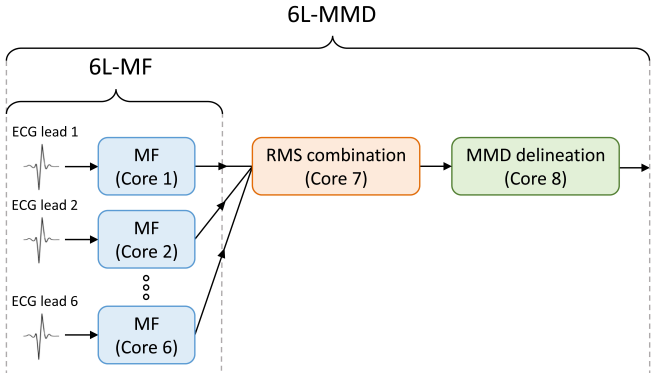

Fig. 7. Task graph of the 6L-MF and 6L-MMD benchmarks.

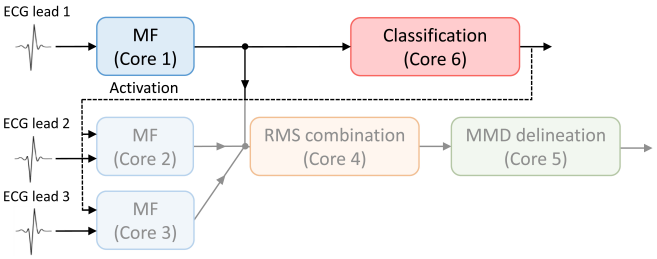

Fig. 8. Task graph of the RP-CLASS benchmark. The shaded part is only activated when an abnormal heartbeat is detected by the classifier.

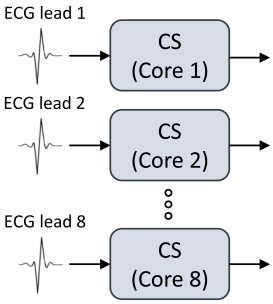

Fig. 9. Task graph of the 8L-CS benchmark.

B. Target and Baseline Systems

Previous studies have showcased that multi-core architectures achieve higher energy efficiency than single-core ones when performing bio-signal applications [16], [50], [54]. Hence, inhere we consider a homogeneous multi-core system

as a baseline solution, and evaluate the further energy and performance gains obtained by the HEAL-WEAR system.

In more detail, we focused on three different architectural configurations, described below.

- *Multi-core architecture only:* No CGRA is present, and the applications are entirely executed by the processors. This system is similar to the one described in [16]. It supports SIMD execution to reduce accesses to the instruction and the data memories. A comparative evaluation with respect to this baseline therefore explores the efficiency of a shared reconfigurable acceleration resource in the WBSN digital signal processing domain.
- *Multi-core architecture with single DP CGRA:* Processors are interfaced with a CGRA having only one DP per RC. Hence, the execution of SIMD kernels is not supported, and SIMD acceleration requests are mapped on different CGRA regions. This setup corresponds to the platform described in [46].
- *HEAL-WEAR – Multi-core architecture with SIMD-CGRA:* Target architecture, as described in Sections III and IV. It presents multiple DPs in each RC, allowing the efficient acceleration of SIMD kernels. We have considered CGRA instances having two or three DPs per RC.

C. Bio-Medical Benchmarks

Experimental evidence was gathered for the above-mentioned platforms when processing ECG records acquired with a sampling rate of 500Hz per lead, common in high-quality ambulatorial recordings [53]. Four complex cardiac processing routines are evaluated as benchmark applications.

- *6L-MF*: Six-lead Morphological Filtering (MF). Removes artifacts (due to muscles activity, system AC supply interferences and base drift caused by breathing) from an ECG acquisition, using structuring elements to remove low- or high-frequency noise components, according to the algorithm described in [55]. This benchmark operates in parallel on six different input streams and is therefore mapped on six cores (cf. Figure 7).
- *6L-MMD*: Six-lead delineation using Multi-scale Morphological Derivatives (MMD) [56]. It detects the characteristic fiducial points (P, Q, R, S and T, as depicted in Figure 10) of each heartbeat. This application relies on 6L-MF to properly filter the acquires signals, then performs a Root-Mean-Square (RMS) combination of the filtered streams, and finally the delineation of the fiducial points. It is divided into three different tasks mapped on eight processing cores, as shown in Figure 7.
- *RP-CLASS*: Uses a heartbeat neuro-fuzzy classifier, operating on a single-lead, to identify pathological heartbeats, by applying a random projection over the heartbeat samples [57]. When a heartbeat abnormality is detected, a three-lead delineation is activated for a short window of signal. RP-CLASS is mapped onto six cores (cf. Figure 8), among which the four cores in the delineation chain are seldom activated.
- *8L-CS*: Eight-lead Compressed Sensing (CS) algorithm [34], mapped onto eight cores (cf. Figure 9), which applies a 50% lossy compression on the input ECG.

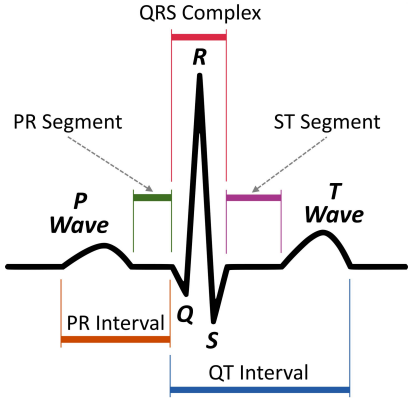

Fig. 10. ECG fiducial points of a normal heartbeat.

The selected applications present different workloads and computational characteristics: the execution of 8L-CS is dominated by a single and intense kernel, which is always executed in SIMD. RP-CLASS, on the other hand, has a more complex run-time behaviour and fewer opportunities for SIMD execution. 6L-MMD and 6L-MF represent a middle ground in-between those two extremes.

D. Accelerated Kernels

The computational kernels of the considered applications were identified with the aid of a profiling pass [58] developed on top of the LLVM compiler infrastructure [59]. Kernels were selected among the most frequently executed basic blocks of the application, discarding candidates which contained constructs that could not be executed on the CGRA, such as dynamic memory allocations. A possible extension of our framework is the full automation of this step, adapting existing instruction set extension identification techniques, such as the ones proposed by [60] and [61]. A pass built on LLVM was also used to derive the Data Flow Graphs (DFGs) of the kernels, which were then modulo-scheduled on the CGRA. In this way, we derived the number of columns and RCs required by the kernels, the operations performed at each clock cycle during a kernel execution, the number of cycles per iteration, and ultimately the content of each configuration word.

Using this strategy, we selected the following kernels:

- *Dbl Min Srch* performs a search of the first and second minimum values inside a window of samples. This kernel is used by the cores running the MF algorithm.
- *Dbl Max Srch*, similarly to *Dbl Min Srch*, performs a search of the first and second maximum values inside a window of samples. This kernel is used by the cores running the MF algorithm.
- *Min Max Srch* determines the minimum and maximum limit values used during the *erosion* and *dilation* steps [55], executed by the cores running the MF algorithm.
- *Sqrt 32* executes a 32 bits square root algorithm. This kernel is used by the core performing the RMS combination of the filtered ECG signals.

TABLE I
KERNELS UTILIZATION PER APPLICATION. EACH CELL INDICATES THE NUMBER OF PROCESSING CORES REQUESTING EACH ACCELERATION.

Kernels \ Apps	6L-MF	6L-MMD	RP-CLASS	8L-CS
Dbl Min Srch	6 cores	6 cores	3 cores	-
Dbl Max Srch	6 cores	6 cores	3 cores	-
Min Max Srch	6 cores	6 cores	3 cores	-
Sqrt 32	-	1 core	1 core	-
Lin Srch	-	1 core	1 core	-
Apply RP	-	-	1 core	-
Lin Min Max	-	-	1 core	-
CS	-	-	-	8 cores

- *Lin Srch* performs a linear search of the two minimum values and two maximum values inside an array of samples. Its execution is more compact than the one of the kernels *Dbl Min Srch* and *Dbl Max Srch* executed sequentially. This kernel is used by the core executing the RMS combination of the filtered ECG signals.
- *Apply RP* calculates the random projection [57] on a signal window, by performing a matrix-vector multiplication. This kernel is used by the core executing the classifier.
- *Lin Min Max* performs a linear search of a single minimum value and single maximum value inside an array of samples. This kernel is used by the core performing the classification task.
- *CS* computes the sequence of random indexes necessary to perform the Compressed Sensing. This kernel is used by the cores running the CS algorithm.

Table I shows, for each application, which kernels are accelerated, and how many cores request them.

VI. EXPERIMENTAL RESULTS

This section showcases the performance of the heterogeneous HEAL-WEAR system in a bottom-up fashion. First, the performance, area and energy-efficiency of the SIMD-CGRA is evaluated when executing individual kernels. Then, a system-level assessment is performed on the overall platform and over all benchmark applications, and is concluded by providing insights on the trade-off implied by different choices for the CGRA SIMD width.

A. CGRA Performance Evaluation

1) Kernels Execution Speed-up: Figure 11 shows the speed-ups of the considered kernels when executed on the CGRA, with respect to their execution time on the processors. They range from 0.8x to 11.3x, depending on the kernel structure (e.g.: number of iterations, initiation interval) as well as the number of DPs embedded in the cells of the CGRA mesh (the CGRA SIMD width). For each kernel, the presented data considers both configuration and execution time, as well as the overhead due to the management of acceleration requests. Since kernels compete for the limited CGRA resources, their

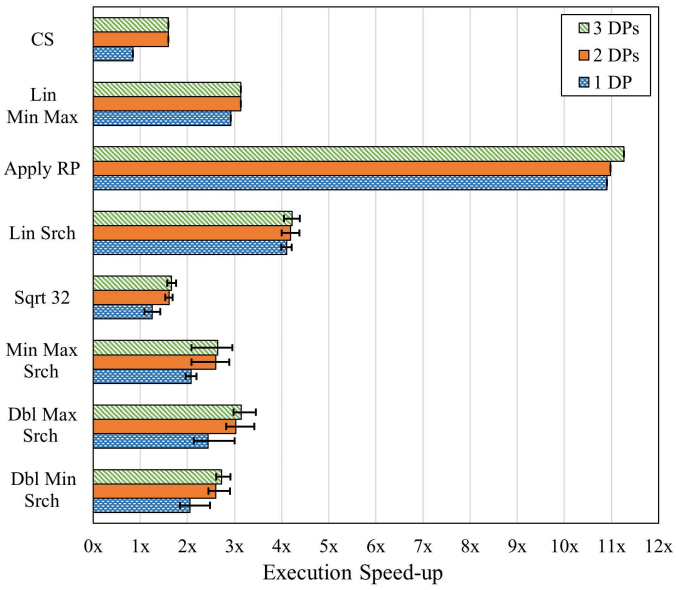

Fig. 11. Average speed-up with kernels running on the multi-core + CGRA platform w.r.t. kernels running only on the multi-core platform.

speedup may vary across invocations. Hence, Figure 11 provides average, maximum and minimum speedup values for the cases having a non-negligible variance.

In all cases, kernels require a smaller execution time on the CGRA than in the processors. The *CS* kernel on a 1 DP CGRA is an outlier, as it has a speedup of 0.8x (a slowdown), due to the high amount of contention during its execution. Such a contention level is substantially reduced by increasing the number of DPs, as SIMD accelerations can be configured and mapped in parallel on the CGRA, ultimately improving run-time performance. In fact, the average execution time decreases for all kernels when a SIMD-CGRA is employed, allowing for instance the *CS* kernel to reach 1.6x speedup when multi-DPs are employed.

2) Resource Utilization Analysis: By employing multi-DP RCs, concurrent SIMD accelerations can be mapped and executed at the same time on the same set of RC columns. To investigate how much the different kernels benefit from SIMD-CGRA, Figure 12 depicts the repartition of the different types of acceleration calls over the execution of the considered benchmarks for single- and multi-DP configurations. In this figure, kernels *Sqrt 32*, *Lin Srch*, *Apply RP* and *Lin Min Max* run on a single core and thus can not take advantage of the SIMD execution mode (cf. Table I in Section V-D). For these cases, Figure 12 only shows one bar that remains identical for the different configurations. On the other hand, kernels *Min Max Srch* and *CS* benefits the most from the SIMD mode, with 100% of the acceleration requests executed in this mode.

In addition to improving the execution speed-up of the kernels and the parallelism on the CGRA mesh, the SIMD execution mode allows also a faster access time to the accelerator. When the cores request a kernel acceleration, a variable waiting time occurs between the moment in which the request is received by the Synchronizer and the moment in which it is transmitted to the accelerator by the CGRA

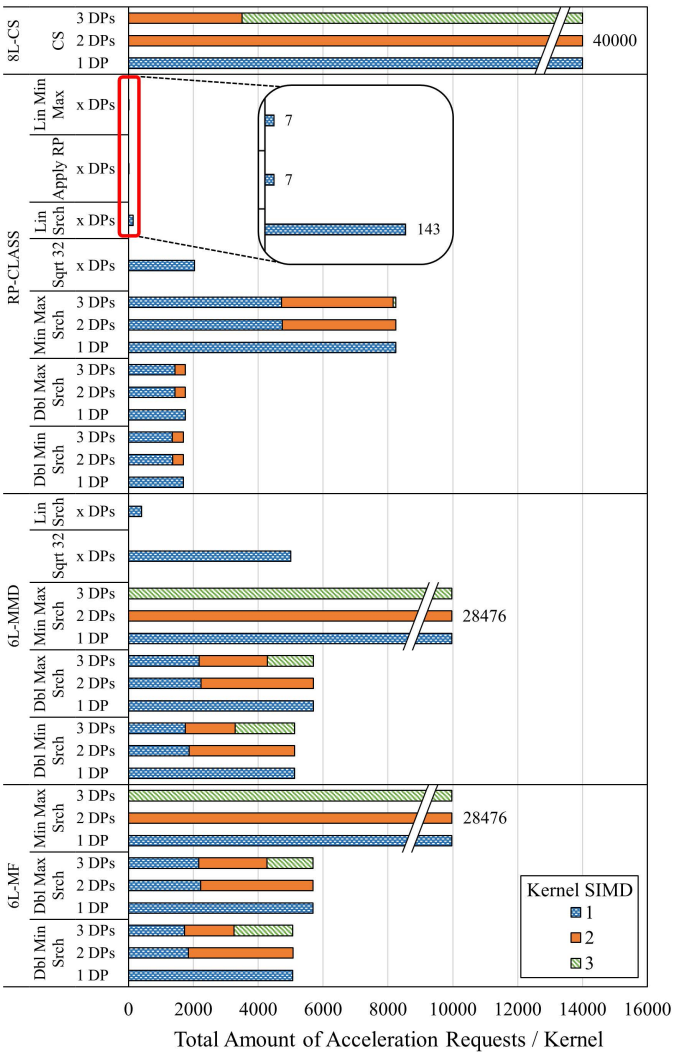

Fig. 12. Repartition of kernel acceleration requests types, depending on employed datapath configuration.

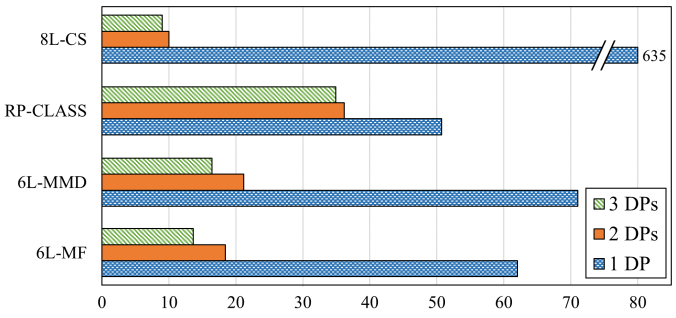

Fig. 13. Average waiting time (in cycles) spent for each acceleration request (called by the cores executing the different benchmarks), before being configured and executed on the CGRA.

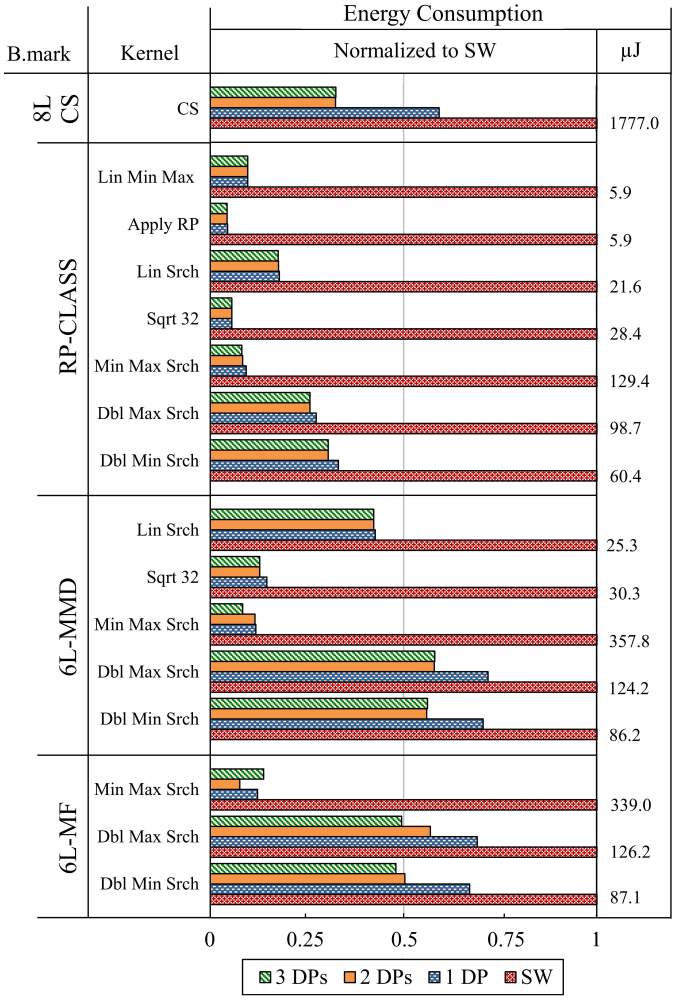

Fig. 14. Energy consumption of the different kernels. For each kernel, the bars are normalized to the SW-only consumption, which is indicated on the right side of the graph.

Controller. This waiting time is depicted on Figure 13 for the different DP configurations. As shown by this figure, there is a sharp drop between the 1 DP case and the other datapath configurations. In fact, the 1 DP configuration induces a large amount of contention on the CGRA, i.e. 17.9x more waiting time in average for all the considered benchmarks with the 1 DP configuration compared to the 2 DP one.

3) Kernels Energy Consumption: Figure 14 compares the total energy consumed when executing the selected kernels in each application, on the multi-core architecture only and on the CGRA with different SIMD widths. The data is aggregated across all invocations of a kernel in the indicated benchmark.

The figure highlights that execution on the CGRA is more energy-efficient than in software by a large margin. Moreover, by supporting SIMD in the CGRA mesh, further energy savings can be achieved, especially for kernels which present a high ratio of concurrent calls by multiple cores *(Dbl Min Srch, Dbl Max Srch, Min Max Srch and CS)* (cf. Figure 12). The average savings relative to these kernels for a CGRA with a SIMD width of 2, averaged over all the applications where the kernels are used, range from 53% (*Dbl Max Srch*) to more than 90.8% (*Min Max Srch*).

4) CGRA Area Footprint Exploration: Table II details the breakdown of the area footprint of the different CGRA components. An important part of the mesh footprint is used by the Configuration RAM, which occupies more than half of the

CGRA real estate in the case of 2 DP instance (which requires roughly $0.5mm^2$). The Configuration RAM size could be reduced by supporting less kernels per application (or smaller ones), or by storing configurations in a compressed form [62]. The mesh itself is extremely compact, as only few operations have to be supported by the ALUs and only a simple control logic is required. While ALUs supporting the square root operation are bigger than the ones who do not, such overhead has a small overall impact, as this feature is present only in one cell per column.

In fact, each DP inside an RC (ALUs + local register file + multiplexers) are smaller than the configuration memory of the RC itself. Since the configuration storage and the control logic are shared among the different DPs, the overhead of doubling the SIMD width from one to two is therefore only 33%.

B. System-Level Assessment

In this section, we investigate the energy benefits, area overhead and run-time performance of HEAL-WEAR.

1) Energy Consumption per Application: Figure 15 analyzes the energy consumed by the multi-core platform with and without the SIMD-CGRA, for the processing of the targeted applications. The processing system including the CGRA accelerator results in significant energy savings for all the considered benchmarks, ranging from 9.2% to 37.2%. The maximum reduction of energy consumption is observed for the 8L-CS application, because its workload is mostly concentrated in the accelerated *CS* kernel. Substantial energy gains are also noted in the case of 6L-MF and 6L-MMD applications, which similarly spend a large amount of their processing time in the accelerated kernels. Even in the case of the RP-CLASS application, where there are far fewer acceleration calls (cf. Figure 12), the obtained energy savings are still significant (9.2%). Furthermore, Figure 15 highlights that using the SIMD-CGRA improves energy efficiency, as the accelerator takes advantage of SIMD processing.

2) Platform Area Breakdown: This section investigates the effects of using the proposed SIMD-CGRA on the area footprint of the whole system. Figure 16 displays the area breakdown of the different components of the envisaged HEAL-WEAR platform. It showcases that the addition of the

TABLE II
CGRA AREA EXPLORATION (IN 65 NM UMC TECHNOLOGY LIBRARY)

Component	Area (μm^2)
Configuration register (per RC)	6721.5
CGRA control	5227.9
Cell control (per RC)	881.7
Register file (per DP)	1374.2
Multiplexers (per DP)	822.9
ALU (per DP)	3962.8
ALU_sqrt (per DP)	7859.1
Configuration_RAM	278978.2
Total CGRA area (1 DP)	**361591.2**
Total CGRA area (2 DPs)	**484900.7**
Total CGRA area (3 DPs)	**593348.6**

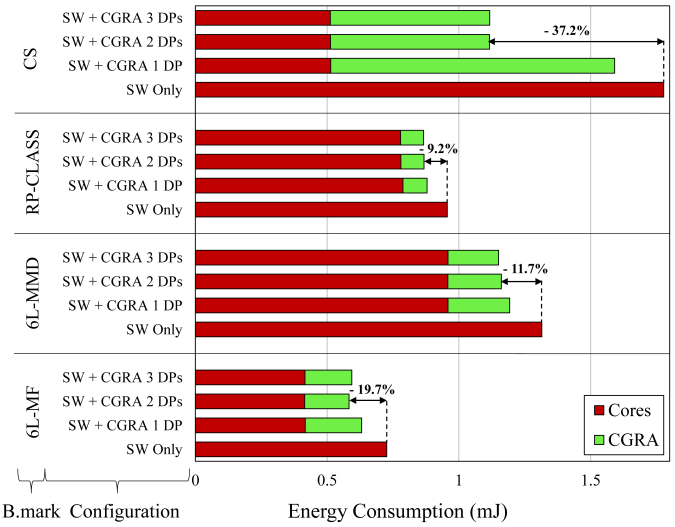

Fig. 15. System energy consumption for processing the different benchmark applications.

CGRA with a single DP results in the increase of the area occupied by only 29.5%, with respect to a multi-core system alone. Moreover, the area cost of extending the CGRA to support SIMD kernels is marginal: less than 9% of the total area for each additional DP. Thus, the area penalty deriving from the adoption of complex multi-DP cells, which achieve a better energy efficiency when the execution of SIMD kernels dominate at run-time (as seen in Section VI-B1), is affordable. For example, in the case of the 8L-CS benchmark, the system energy consumption is 29.9% lower when a CGRA with 2 DPs per RC is used instead of a single-DP CGRA. In that case, the system area penalty incurred due to the additional DP is only 5.8%.

3) CGRA SIMD width: From a design perspective, the choice of which SIMD width to support is a trade-off between complexity and flexibility. A higher SIMD degree can potentially decrease energy consumption by harnessing more

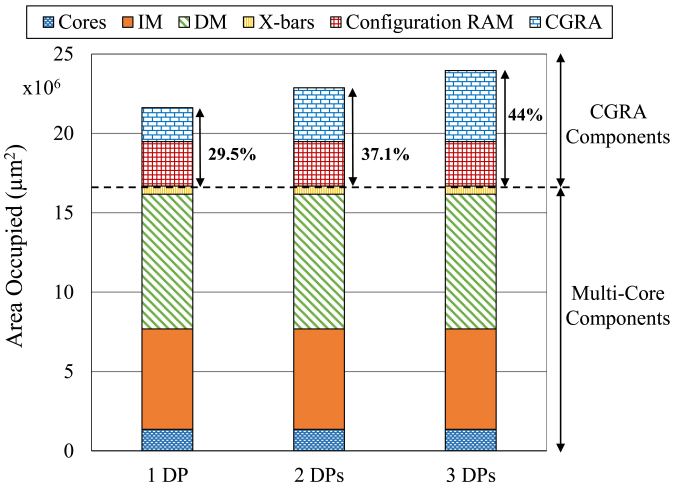

Fig. 16. Area breakdown of HEAL-WEAR platforms, embedding a CGRA with different SIMD widths.

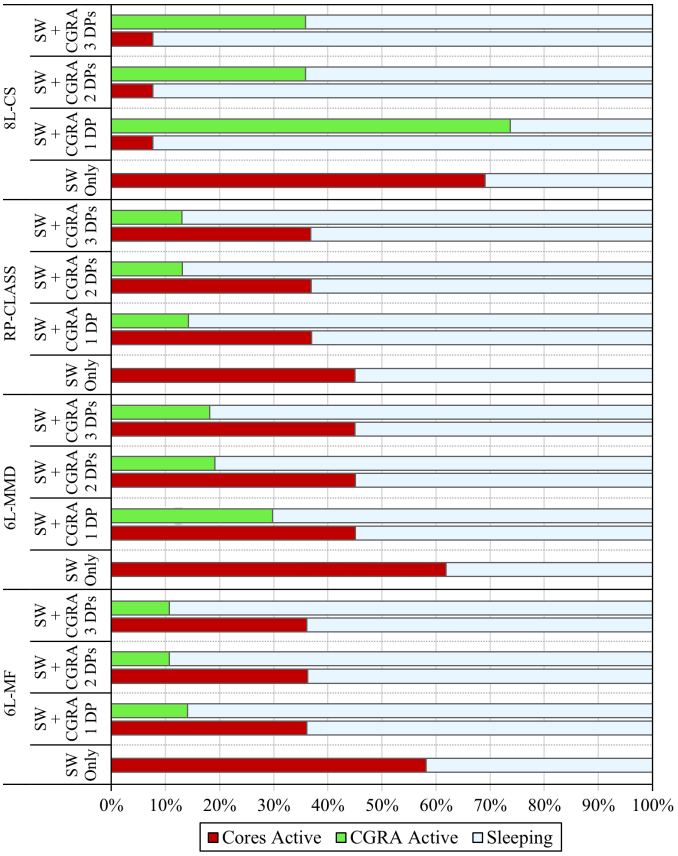

Fig. 17. Multi-core and CGRA utilization time in the considered platforms (% of the total run-time).

parallelism from applications. On the other hand, the design of more complex RCs require more silicon area, and a more elaborate controller. Ultimately, the parallelism supported by the accelerator should closely match the one present in the kernels of application.

A configuration with a single DP per cell is clearly sub-optimal. As discussed in Section VI-A1, in the case of the example *CS* kernel, the adoption of a single-DP mesh even results in a slowdown (instead of a speedup), due to the presence of a high level of contention, which leads to significant waiting times (reported in Section VI-A2). Ultimately, as detailed in Section VI-A3, the energy efficiency of a single-datapath CGRA trails that of an alternative supporting SIMD, Figure 15 shows that the energy savings when using a CGRA with 2 DPs per RC are very similar to the case when 3 DPs are employed, while requiring a bigger area (Section VI-A4). Therefore, a choice of 2 DPs per RC strikes a good trade-off between performance, energy efficiency and silicon area for the considered benchmarks.

We highlight the good performance of this configuration by analyzing in Figure 17 the utilization of its computing components. Results show that, when the system is interfaced with a 2 DPs SIMD-CGRA, the active time of the cores is decreased on average from 58.7% to 31.6%, compared to the software-only version. This reduction is instrumental in improving the overall power consumption of the platform, because of the superior energy efficiency of the CGRA module. Moreover, for all the studied benchmarks, a SIMD-CGRA with 3 DPs only provides marginal benefits with respect to the 2 DPs configuration, justifying the choice of the SIMD-CGRA employing 2 DPs per RC. Finally, Figure 17 also shows that in the case of the computationally intensive 8L-CS benchmark, the overall utilization of the multi-core system and the CGRA is below 50% when employing a multi-DP configuration, allowing a potential reduction of the operating frequency from 2 MHz to 1 MHz.

VII. Conclusion

Energy efficiency is a major concern in the design of digital systems across the computing landscape. It is especially important in the context of Internet of Things (IoT) appliances, where the power budget is drastically limited. To achieve the required ultra-low-power operating levels, a careful optimization of the architectural components is mandatory. For such optimization to be effective, it must be domain-specific, i.e.: it has to take into account and exploit the characteristics of target workloads. Herein, we addressed this complex endeavour by proposing the HEAL-WEAR platform, a heterogeneous architecture devoted to bio-signal processing applications.

In this domain, the run-time execution profile of applications is often divided between control-dominated phases and computationally-intensive ones, presenting with compact loops (kernels). The illustrated HEAL-WEAR platform can efficiently support both: the former on multiple ultra-low power processing cores, the latter by employing a coarse-grained reconfigurable array interfaced to the cores as a shared acceleration resource. Moreover, bio-signal analysis applications often process multiple input sources in parallel. We leverage this characteristic as an energy-saving feature by supporting SIMD modes in the processor as well as in CGRA. The resource management of HEAL-WEAR is performed with a low-overhead strategy, based on a dedicated Instruction Set Extension, which is interpreted by a lightweight hardware synchronizer to orchestrate the run-time execution of bio-signal processing applications.

The above-mentioned features allow the HEAL-WEAR platform to achieve tangible energy savings of up to 37.2% when executing complex ECG processing applications, in comparison to an equivalent multi-core solution that does not feature hardware acceleration.

Acknowledgment

This work has been partially supported by the E4Bio (grant no. 200021 159853) RTD project evaluated by the Swiss NSF, as well as by the BodyPoweredSenSE (grant no. 20NA21 143069) RTD project evaluated by the Swiss NSF and funded by Nano-Tera.ch with Swiss Confederation financing.

References

[1] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, April 2011.

[2] MEP Heart Group., "Cardiovascular diseases facts and figures," http://www.mepheartgroup.eu/index.php/facts-a-figures, August 2016.

[3] World Health Organization., "Cardiovascular diseases," http://www.who.int/topics/cardiovascular_diseases/en, August 2016.

[4] P. Bonato, "Wearable sensors and systems," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, no. 3, pp. 25–36, May 2010.

[5] R. Vecht et al., "ECG diagnosis in clinical practice." London, UK: Springer-Verlag, 2009.

[6] F. Zhang et al., "Design of ultra-low power biopotential amplifiers for biosignal acquisition applications," *IEEE Transactions on Biomedical Circuits and Systems (TBioCAS)*, vol. 6, no. 4, pp. 344–355, August 2012.

[7] R. Braojos et al., "Ultra-low power design of wearable cardiac monitoring systems," in *51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2014, pp. 1–6.

[8] F. Chen et al., "Energy-aware design of compressed sensing systems for wireless sensors under performance and reliability constraints," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 3, pp. 650–661, March 2013.

[9] H. Mamaghanian et al., "Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, Sept 2011.

[10] D. Daly et al., "A 6-bit, 0.2 V to 0.9 V highly digital flash ADC with comparator redundancy," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 11, pp. 3030–3038, Nov 2009.

[11] E. Nemati et al., "A wireless wearable ECG sensor for long-term applications," *IEEE Communications Magazine*, vol. 50, no. 1, pp. 36–43, January 2012.

[12] Texas Instruments, S. Kamath and J. Lindh, "Measuring Bluetooth Low Energy Power Consumption," http://www.ti.com/lit/an/swra347a/swra347a.pdf, 2016.

[13] R. Braojos et al., "Nano-engineered architectures for ultra-low power wireless body sensor nodes," in *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2016, pp. 1–10.

[14] R. H. Dennard et al., "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 9, no. 5, pp. 256–268, October 1974.

[15] M. Ashouei et al., "A voltage-scalable biomedical signal processor running ECG using 13pJ/cycle at 1MHz and 0.4V," in *IEEE International Solid-State Circuits Conference (ISSCC)*, February 2011, pp. 332–334.

[16] R. Braojos et al., "Hardware/software approach for code synchronization in low-power multi-core sensor nodes," in *Design, Automation Test in Europe Conference Exhibition (DATE)*. IEEE, March 2014, pp. 1–6.

[17] D. Bortolotti et al., "Rakeness-based compressed sensing on ultra-low power multi-core biomedicai processors," in *Proceedings of the 2014 Conference on Design and Architectures for Signal and Image Processing*, Oct 2014, pp. 1–8.

[18] I. Al Khatib et al., "MPSoC ECG biochip: A multiprocessor system-on-chip for real-time human heart monitoring and analysis," in *Proceedings of the 3rd Conference on Computing Frontiers*. New York, NY, USA: ACM, 2006, pp. 21–28.

[19] V. Annese et al., "A digital processor architecture for combined EEG/EMG falling risk prediction," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 714–719.

[20] O. Bai et al., "A wireless, smart EEG system for volitional control of lower-limb prosthesis," in *TENCON 2015 - 2015 IEEE Region 10 Conference*, Nov 2015, pp. 1–6.

[21] N. Sarkany et al., "The design of a mobile multi-channel bio-signal measuring system for rehabilitation purposes," in *2014 14th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, July 2014, pp. 1–2.

[22] L. Ming-hau et al., "Design and implementation of the morphosys reconfigurable computing processor," *Journal of Signal Processing Systems*, vol. 24, no. 2-3, pp. 147–164, March 2000.

[23] N. Ozaki et al., "Cool Mega-Arrays: Ultralow-power reconfigurable accelerator chips," *IEEE Micro*, vol. 31, no. 6, pp. 6–18, Nov 2011.

[24] I. Kuon et al., "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 26, no. 2, pp. 203–215, February 2007.

[25] Y. Hao et al., "Wireless body sensor networks for health-monitoring applications," *Physiological measurement*, vol. 29, no. 11, p. R27, November 2008.

[26] A. Pantelopoulos et al., "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part C Applications and Reviews*, vol. 40, no. 1, pp. 1–12, October 2010.

[27] R. Fensli et al., "A wearable ECG-recording system for continuous arrhythmia monitoring in a wireless tele-home-care situation," in *18th IEEE Symposium on Computer-Based Medical Systems (CBMS)*, June 2005, pp. 407–412.

[28] N. Oliver et al., "HealthGear: a real-time wearable system for monitoring and analyzing physiological signals," in *IEEE International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, April 2006, pp. 1–4.

[29] M. Sung et al., "Wearable feedback systems for rehabilitation," *Journal of NeuroEngineering and Rehabilitation (JNER)*, vol. 2, no. 1, p. 1, June 2005.

[30] M. Seok et al., "The phoenix processor: A 30pW platform for sensor applications," in *IEEE Symposium on VLSI Circuits (VLSIC)*, June 2008, pp. 188–189.

[31] S. R. Sridhara et al., "Microwatt embedded processor platform for medical system-on-chip applications," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 46, no. 4, pp. 721–730, April 2011.

[32] Y. He et al., "Xetal-Pro: An ultra-low energy and high throughput SIMD processor," in *47th ACM/IEEE Design Automation Conference (DAC)*, June 2010, pp. 543–548.

[33] A. Y. Dogan et al., "Power/performance exploration of single-core and multi-core processor approaches for biomedical signal processing," in *Proceedings of the 21st International Conference on Integrated Circuit and System Design: Power and Timing Modeling, Optimization, and Simulation*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 102–111.

[34] J. Constantin et al., "TamaRISC-CS: An ultra-low-power application-specific processor for compressed sensing," in *IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, October 2012, pp. 159–164.

[35] J. Kwong et al., "An energy-efficient biomedical signal processing platform," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 46, no. 7, pp. 1742–1753, July 2011.

[36] F. Conti et al., "Synthesis-friendly techniques for tightly-coupled integration of hardware accelerators into shared-memory multi-core clusters," in *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. IEEE, September 2013, pp. 1–10.

[37] P. Garcia et al., "Kernel sharing on reconfigurable multiprocessor systems," in *International Conference on Electrical and Computer Engineering (ICECE)*. IEEE, December 2008, pp. 225–232.

[38] F. Bouwens et al., "Architectural exploration of the ADRES coarse-grained reconfigurable array," in *Reconfigurable Computing: Architectures, Tools and Applications*. Springer, March 2007, pp. 1–13.

[39] K. Patel et al., "Coarse-grained reconfigurable array based architecture for low power real-time seizure detection," *Journal of Signal Processing Systems*, vol. 82, no. 1, pp. 55–68, 2016.

[40] B. Mei et al., "Exploiting loop-level parallelism on coarse-grained reconfigurable architectures using modulo scheduling," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2003, pp. 296–301.

[41] T. Peyret et al., "An automated design approach to map applications on CGRAs," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, May 2014, pp. 229–230.

[42] G. Ansaloni et al, "Slack-aware scheduling on coarse-grained reconfigurable arrays," in *Design, Automation Test in Europe Conference Exhibition (DATE)*. IEEE, March 2011, pp. 1–4.

[43] P. Theocharis et al., "A bimodal scheduler for coarse-grained reconfigurable arrays," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 2, pp. 15:1–15:26, Jun. 2016.

[44] H. Park et al., "Edge-centric modulo scheduling for coarse-grained reconfigurable architectures," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*. New York, NY, USA: ACM, 2008, pp. 166–176.

[45] L. Chen et al., "Shared reconfigurable fabric for multi-core customization," in *48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2011, pp. 830–835.

[46] L. Duch et al., "A multi-core reconfigurable architecture for ultra-low power bio-signal analysis," in *IEEE Biomedical Circuits and Systems (BioCAS)*, October 2016, pp. 1–4.

[47] B. De Sutter et al., *Handbook of Signal Processing Systems*. Springer, July 2010, ch. Coarse-Grained Reconfigurable Array Architectures.

[48] G. Ansaloni et al., "EGRA: A coarse-grained reconfigurable architectural template," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 6, pp. 1062–1074, June 2011.

[49] B. R. Rau, "Iterative module scheduling: an algorithm for software pipelining loops," in *Microarchitecture, 1994. MICRO-27. Proceedings of the 27th Annual International Symposium on*, November 1994, pp. 63–74.
[50] A. Y. Dogan et al., "Synchronizing code execution on ultra-low-power embedded multi-channel signal analysis platforms," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 396–399.
[51] "ASIP Designer," https://www.synopsys.com/dw/ipdir.php?ds=asip-designer, 2017.
[52] ARM, "Cortex-M0 Processor," https://www.arm.com/products/processors/cortex-m/cortex-m0.php, 2017.
[53] "PhysioBank," http://www.physionet.org/physiobank/, August 2016.
[54] A. Y. Dogan et al., "Low-power processor architecture exploration for online biomedical signal analysis," *IET Circuits, Devices Systems (IET-CDS)*, vol. 6, no. 5, pp. 279–286, September 2012.
[55] Y. Sun et al., "ECG signal conditioning by morphological filtering," *Computers in Biology and Medicine (CBM)*, vol. 32, no. 6, pp. 465–479, November 2002.
[56] F. Rincon et al., "Development and evaluation of multilead wavelet-based ECG delineation algorithms for embedded wireless sensor nodes," *IEEE Transactions on Information Technology in Biomedicine (T-ITB)*, vol. 15, no. 6, pp. 854–863, November 2011.
[57] R. Braojos et al., "A methodology for embedded classification of heartbeats using random projections," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 899–904.
[58] G. Zacharopoulos and L. Pozzi, "ClrFreqCFGPrinter: A tool for frequency annotated control flow graph generation," in *European LLVM Developers Meeting*, March 2017.
[59] C. Lattner et al., "LLVM: A compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization (CGO)*, March 2004.
[60] J. Cong et al., "Application-specific instruction generation for configurable processor architectures," in *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*. New York, NY, USA: ACM, 2004, pp. 183–189.
[61] L. Pozzi et. al, "Exact and approximate algorithms for the extension of embedded processor instruction sets," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1209–29, Jul. 2006.
[62] B. Egger et al., "A space- and energy-efficient code compression/decompression technique for coarse-grained reconfigurable architectures," in *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, Feb 2017, pp. 197–209.

Loris Duch received the M.Sc. degree in Engineering, Physics, Electronics and Materials, specialized in digital integrated circuit design from the Grenoble Institute of Technology (Grenoble INP), France, in 2014. He is currently working toward the Ph.D. degree in Microsystems and Microelectronics at the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. His main research interests include: hardware and software co-design exploration, multi-core architectures, energy vs. reliability aware circuit optimization and ultra-low power integrated circuit design, in the field of bio-medical signal processing.

Soumya Basu is a Ph.D. student at the Embedded Systems Laboratory at the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. He received his Masters degree in 2013 from the Politecnico di Torino, Italy, specializing in embedded systems. His research interests lie in ultra-low power hardware-software co-design methods, mainly focusing on reconfigurable architectures for bio-medical signal processing. His parallel research interest is on the application of energy-aware inexact computation methods in the afore-mentioned domain.

Rubén Braojos is currently a post-doctoral researcher at the Embedded Systems Laboratory (ESL) at the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, where he obtained his Ph.D. degree in electrical engineering in 2016. He received his B.Sc and M.Sc. degrees in computer science and engineering from Complutense University of Madrid (UCM), Spain, in 2008 and 2010. His research interests include embedded bio-signal processing, low-power architectures for embedded systems, hardware/software co-design methodologies for energy efficiency and Wireless Body Sensor Nodes applied to the field of healthcare monitoring.

Giovanni Ansaloni is currently a post-doctoral researcher at the Faculty of Informatics of Universitá della Svizzera Italiana (USI-Lugano, Switzerland). From 2011 to 2015, he was a researcher at EPFL (Lausanne, Switzerland). He received the M.Sc. degree in Electronic Engineering from University of Ferrara (Italy) in 2003, the MAS degree from the ALaRI institute (Switzerland) in 2005 and the Ph.D. Degree from University of Lugano (Switzerland) in 2011. His research efforts focus on smart Wireless Body Sensor Nodes systems and applications, including software optimizations of processing algorithms for bio-signal analysis and architectural explorations of ultra-low-power WBSN platforms.

Laura Pozzi received a Ph.D. degree in computer engineering from Politecnico di Milano, Italy, in 2000, and she is currently a Full Professor at the Faculty of Informatics, University of Lugano (USI), Switzerland. Previously she was a post-doctoral researcher at EPFL, a research engineer with STMicroelectronics in California, and an Industrial Visitor at UC Berkeley. Prof. Pozzi has served as Associate Editor of the IEEE Transactions on Computer-Aided Design (TCAD) and IEEE Design and Test, and is in the Technical Program Committee of several international conferences in the areas of Compilers and Architectures for Embedded Systems. Her research interests include automating embedded processor customisation, high performance compiler techniques, innovative reconfigurable fabrics, and, more recently, computational biology.

David Atienza (M'05-SM'13-F'16) is associate professor of electrical and computer engineering, and director of the Embedded Systems Laboratory (ESL) at the Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland. He received his Ph.D. in computer science and engineering from UCM, Spain, and IMEC, Belgium, in 2005. His research interests include system-level design and thermal-aware optimization methodologies for 2D/3D high-performance multi-processor system-on-chip (MPSoC) and ultra-low power system architectures for wireless body sensor nodes. He is a co-author of more than 250 papers in peer-reviewed international journals and conferences, several book chapters, and seven patents. Dr. Atienza received an ERC Consolidator Grant in 2016, the IEEE CEDA Early Career Award in 2013, the ACM SIGDA Outstanding New Faculty Award in 2012, and a Faculty Award from Sun Labs at Oracle in 2011. He served as DATE 2015 Program Chair and DATE 2017 General Chair. He is a Senior Member of ACM and an IEEE Fellow.