Journal of
*Low Power Electronics and Applications*

MDPI

*Article*

# Flexible, Scalable and Energy Efficient Bio-Signals Processing on the PULP Platform: A Case Study on Seizure Detection†

**Fabio Montagna *, Simone Benatti and Davide Rossi**

Energy Efficient Embedded Systems (EEES) Lab - DEI, Viale Risorgimento 2, University of Bologna, 40136 Bologna, Italy; simone.benatti@unibo.it (S.B.); davide.rossi@unibo.it (D.R.)

* Correspondence: fabio.montagna@unibo.it; Tel./Fax: +39-051-2093835

† This paper is an extended version of the paper [1] entitled Sub-pJ per operation scalable computing: The PULP experience. In Proceedings of the 2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), Burlingame, CA, USA, 10–13 October 2016; pp. 1–3.

Academic Editors: David Bol and Steven Vitale
Received: 1 March 2017; Accepted: 6 June 2017; Published: 11 June 2017

**Abstract:** Ultra-low power operation and extreme energy efficiency are strong requirements for a number of high-growth application areas requiring near-sensor processing, including elaboration of biosignals. Parallel near-threshold computing is emerging as an approach to achieve significant improvements in energy efficiency while overcoming the performance degradation typical of low-voltage operations. In this paper, we demonstrate the capabilities of the PULP (Parallel Ultra-Low Power) platform on an algorithm for seizure detection, representative of a wide range of EEG signal processing applications. Starting from the 28-nm FD-SOI (Fully Depleted Silicon On Insulator) technology implementation of the third embodiment of the PULP architecture, we analyze the energy-efficient implementation of the seizure detection algorithm on PULP. The proposed parallel implementation exploits the dynamic voltage and frequency scaling capabilities, as well as the embedded power knobs of the PULP platform, reducing energy consumption for a seizure detection by up to 10× with respect to a sequential implementation at the nominal supply voltage and by 4.2× with respect to a sequential implementation with voltage scaling. Moreover, we analyze the trans-precision optimization of the algorithm on PULP, by means of a hybrid fixed- and floating-point implementation. This approach reduces the energy consumption by up to 43% with respect to the plain fixed-point and floating-point implementations, leveraging the requirements in terms of the precision of the kernels composing the processing chain to improve energy efficiency. Thanks to the proposed architecture and system-level approach for optimization, we demonstrate that PULP reduces energy consumption by up to 140× with respect to commercial low-power microcontrollers, being able to satisfy the real-time constraints typical of bio-medical applications, breaking the barrier of microwatts for a 50-ms complete seizure detection and a few milliwatts for a 5-ms detection latency on a fully-programmable architecture.

**Keywords:** near-threshold computing; parallel architectures; EEG signal processing; seizure detection; trans-precision computing

## 1. Introduction

Recently, several neural diseases are treated through systems that are becoming increasingly effective thanks to the progresses in the field of Brain Machine Interfaces (BMI). Approximately one billion people are affected by severe and disabling neurological pathologies [2]. Among them, epilepsy is characterized by recurrent seizures caused by abnormal neuronal electrical activity, which

can lead to convulsions and loss of consciousness. Over the past few decades, drug delivery and brain surgery were largely used in treating epileptic seizures even though they present several drawbacks, like drug toxicity and risk connected to head surgery. Furthermore, there are at least 25% of epileptic subjects that present resistance to drugs or with surgically intractable seizures. Neuromodulation [3] is a brain stimulation technique based on the injection of small currents directly on the neural tissues. It has been demonstrated that neuromodulation has significant benefits on treating epilepsy while it does not present collateral effects [4]. Former neuromodulation devices were designed to give a continuous constant stimulation to the brain tissues, but they lack battery lifetime and efficient delivery of the treatment. Nowadays, to optimize the stimulation and enhance the battery life of the neuromodulators, closed loop real-time systems are gaining ground [5]. Electroencephalogram (EEG) analysis allows monitoring neural activity maintaining a low level of obtrusiveness, since EEG traces are acquired directly placing electrodes on the scalp of the subject. A seizure can be identified from the EEG trace as an unexpected change in the amplitude and the frequency of the neural signal; hence the need to develop a responsive closed-loop neuromodulation system, which is able to detect seizures in a fully-automatic fashion, enabling a direct electrical feedback only when required.

Neuromodulation systems typically implement algorithms that analyze the EEG signal to detect changes representing a seizure activity. While the trend in research goes toward the design of dense multichannel systems [6–8] with large and dense arrays of sensors (i.e., up to 128 electrodes) to allow a fine-grain coverage of the brain surface and target wider areas, current wearable and implantable solutions for seizure detection are only able to manage a few electrodes with a latency of 500 ms due to their limited computing power [9,10]. The most common deeply-embedded systems for seizure detection are based on machine learning that can achieve high accuracy with a relatively small computational effort, for a limited number of electrodes. Most of these systems are based on fixed-functions ASIC designs, implementing feature extraction and pattern recognition algorithms for classification [11,12]. The main issue with machine learning approaches such as SVM is that the computational complexity significantly increases with the number of channels, making it challenging to implement these algorithms in deeply-embedded systems [13]. From the algorithmic point of view, this issue has been addressed applying dimensionality reduction algorithms, such as PCAs, at the beginning of the processing chain. Although this approach has been exploited in several works elaborating the data offline [7], embedded solutions working with a number of electrodes greater than 18 have not been presented, yet. Since the computational requirements for these algorithms are challenging and the complexity scales up with the number of sensors, the design of a scalable digital architecture must target energy efficiency for a wide range of workloads.

The open challenge we address in this work is to design an efficient programmable framework for seizure detection, based on the Parallel Ultra-Low Power (PULP) platform, a fully-programmable, scalable and energy-efficient multi-core architecture for sub-mW, deeply-embedded applications. The PULP platform leverages near-threshold operation [14] and multi-core parallelism with explicitly-managed shared L1 memory to overcome performance degradation at low voltage, while maintaining the flexibility and programmability typical of instruction processors. Moreover, it exploits technology-aware architectural optimizations and power management techniques reducing the intrinsic overheads typical of parallel computing platforms, with the goal of improving energy efficiency during the execution of complex signal processing applications. Our recent results with the PULP computing platform demonstrate that pJ per operation (GOPS/mW) computational efficiency is within reach in today's 28-nm Ultra Thin Box and Body (UTBB) Fully Depleted Silicon On Insulator (FD-SOI) technology [1,15].

We show that our platform is able to compute a seizure detection on 23 electrodes in less than 5 ms, improving the state of the art of commercial systems by more than 5× if we compare the number of electrodes, and by 100× in terms of detection latency. We analyze the performance/accuracy trade-off on the hardware/software platform, showing that, even if a fixed-point implementation of the application provides lower power consumption over floating-point operations, the latter provides

a smaller energy for applications requiring high dynamic range and precision such as EEG processing, especially for algorithms requiring iterative methods [16]. We therefore propose a trans-precision implementation hardware/software computing approach that introduces strong approximations in intermediate computation, while safely keeping the error in the final result bounded, which mixes floating-point and fixed-point computations, improving the energy efficiency of the system by 1.5× with respect to the plain fixed-point implementation and by 1.4× with respect to a pure floating-point implementation, with a loss of precision bounded to only 3.5% with respect to our reference golden model represented by the double precision floating-point implementation of the algorithm in MATLAB. Power management techniques embedded in the PULP platform further improve energy efficiency by 1.4×. Furthermore, we compare the results with low-power commercial microcontrollers based on 32-bit ARM Cortex M4 processors showing that, by virtue of our energy-efficient architecture and of the optimizations of the algorithmic implementation, we reduce the energy required to compute the algorithm up to 140×, depending on the detection latency requirements. As opposed to ASIC solutions with hardwired functions traditionally employed in embedded biomedical applications, the proposed computational framework based on a fully software programmable multi-core architecture is highly scalable, versatile and can be used for a wide range of applications.

The paper is organized as follows: Section 2 provides an overview of the state of the art of ULP architectures and of brain machine interface applications. Sections 3 and 4 present an in-depth description of the PULP platform and seizure detection algorithm, emphasizing optimization strategies targeting the implementation on PULP such as parallelization schemes, floating- to fixed-point conversion (with a comparison in term of loss of precision, accuracy in the classification stage and power consumption) and power management techniques. In Section 5, we discuss the implementation and profiling of a complete seizure detection algorithm on PULP and on two commercial off-the-shelf ARM Cortex M4-based MCUs, showing the benefits gained by our target platform in terms of execution time and power consumption.

## 2. Related Work

A typical implementation of a BMI includes five main stages: measure of the brain activity, pre-processing, feature extraction, classification and, finally, control stage [17]. The key elements for a seizure detection BMI implementation are the algorithm, which has to be able to detect ongoing seizures with high accuracy and satisfying the real-time constraint, and the architecture, which has to guarantee high computational effort maintaining low energy consumption. Nowadays, the research goes towards the development of multi-channel systems, increasing the number of electrodes placed on the scalp, to obtain a complete coverage of the entire brain surface. This leads to a conspicuous amount of data, which is unsuitable for battery-powered wearable or implantable systems. To cope with these dense arrays of sensors, dimensionality reduction algorithms represent a method to optimize system resources and reduce memory requirements. In [6], the authors describe a wavelet framework for automatic classification of epileptic activity using Principal Component Analysis (PCA). In this work, a dataset collected by researchers of the Bonn University containing data acquired from 128 channels is used to test and validate the application. A three-level Wavelet Packet Decomposition (WPD) is performed, and then, to reduce the dimensionality of the input matrix, PCA is used to select a set of features that contains useful information discarding redundant data. After feature extraction, feature vectors are classified using six different classifiers: Decision Tree (DT), K-Nearest Neighbor (KNN), Gaussian Mixture Model (GMM), Radial Basis Probabilistic Neural Network (RBPNN), etc. Training and testing datasets are selected through a 10-fold cross-validation, and they obtain the best classification accuracy (i.e., 99%) with the GMM classifier. In [7], the authors show a comparison between different approaches for dimensionality reduction (PCA, ICA and LDA). Starting from a 128-channel EEG dataset, they extract frequency information with five-level DWT, and then, they perform the dimensionality reduction stage. After feature extraction, SVM is used to classify data into the seizure or no seizure class. The best performance is obtained in the LDA feature extraction case. The work in [8] shows a method

to recognize epileptic activity through a feature extraction approach based on PCA to reduce the dimension of the data, DWT to extract frequencies at a certain resolution level and ApEn to estimate the quantity of entropy contained in the signal (a low value indicates determinism, while a high value expresses randomness). After this, a threshold is established to be applied to the ApEn values. In this way, it is possible to determine if data are related to a normal EEG activity or to an epileptic seizure. All of the aforementioned approaches show the advantages of dimensionality reduction, even though their presented implementations are tested on desktop devices without the limitations dictated by the use of a fully-embedded architecture.

Other studies are based on the implementation of a seizure detection application on SoC. To the best of our knowledge, all of these works make use of hardware accelerators and/or external computational devices to execute all of the processing chain using a reduced number of electrodes during the acquisition of the EEG signals. The work presented in [11] proposes an SoC for seizure detection based on an Analog Front-End (AFE) with eight channels for EEG acquisition and a digital accelerator for feature extraction and classification. This system acquires the EEG signals, extracts energy features from the eight channels and performs a linear SVM classification to detect the onset of an epileptic seizure. The approach reaches real-time efficient classification, but the system is limited to the eight channels while the HW accelerators are fully hardwired, without any degree of freedom in the processing. Hence, the solution is not scalable and does not meet the trend of increasing the number of channels for the BMI systems [18]. Another similar work is shown in [19]: an eight-channel closed-loop SoC is developed and tested in vivo to contrast the effect of epileptic seizures. Intracranial EEG (iEEG) signals are acquired with eight energy-efficient AFE and processed by the bio-signal processor, which includes a power-efficient FFT core and an ApEn encoder for feature extraction, and seizures are detected using an Linear Least Squares (LLS) classifier. After the output of the classifier, if the onset of an epileptic seizure is detected, an electrical stimulus is generated from an adaptive high-voltage-tolerant stimulator. In [20], a real-time seizure detection for ambulatory EEG signals is presented. In this work, the algorithm performs a pre-processing stage for artifact removal using a band pass FIR filter. Then, features are extracted using five features in the time domain (RMS, number of maxima and minima, line length, nonlinear energy). After feature extraction, the feature vectors are classified testing different types of classifiers among which are LDA and SVM. The processing chain was firstly simulated on a DSP chip and then on an ASIC obtaining a power consumption gain around 25% w.r.t. the first solution. In [21], the authors reported a SoC that acquires up to 18-channel EEG signals with an instrumental amplifier and an ADC performs an on-chip feature extraction, and then, through a low-power parallel-serial interfaces, the feature vector is streamed to a central device where an SVM is employed for the classification.

Some other solutions are oriented toward keeping the flexibility of a programmable platform with some accelerated kernels. The key point of these approaches is that many algorithms share several signal processing kernels (e.g., matrix multiplications, FFT, SVD, etc.). Hence, the attempt to combine the general-purpose instruction set of a programmable platform with a dedicated accelerator aims to strike a balance between energy, flexibility and computational performance. In [12], the authors present a mixed architecture based on the combination of a low power MSP430 microcontroller and a hardware accelerator for machine learning. The configurability of the algorithm kernels allows using the accelerator for logistic regression and SVM with polynomial and RBF kernels. The proposed system was tested on an epileptic seizure detection algorithm reaching sub-mW power consumption. Another example of this kind of specialized architectures is presented in [22], where a 16-bit microcontroller [23] with an RISC architecture based on the standard MSP430 ISA (Instruction Set Architecture) is coupled with a COordinate Rotation Digital Computer (CORDIC) [24,25] accelerator. The system works at 0.5–1 V and, using voltage scaling and block level power optimization, achieves more than a 10× energy reduction with respect to a conventional CPU. Nevertheless, these applications are strongly connected to their target application; they lack scalability; and they are not fully reusable in other application scenarios. In Table 1, the most relevant characteristics are collected allowing a schematic

comparison between these solutions. It is possible to note that current embedded implementations are capable of managing up to 18 electrodes, leveraging fixed function ASICs or highly specialized architectures. It is worth noticing that the authors in [21] claim that it is possible to employ up to 18 EEG channels for the seizure detection application, but as the execution of the SVM on the SoC would lead to a significant increase of the system power, the classification is performed remotely; while in [12], an inflexible SVM hardware accelerator has been employed to allow real-time operation within the power budget.

**Table 1.** Comparison with the state of the art of seizure detection implementation.

| | **Yoo J.** [11] | **Chen W.M.** [19] | **Patel K.** [20] | **Verma N.** [21] | **Lee K.H.** [12] | **Proposed Work** |
|---|---|---|---|---|---|---|
| Applications: | EEG | EEG | EEG | EEG | EEG, ECG | EEG |
| Processing Chain: | spectral energy, SVM | FFT ApEn LLS | FIR, RMS, maxima&minima, line length, nonlinear energy, LDA | spectral energy, SVM | spectral energy, variance, SVM | PCA, DWT + energy, SVM |
| Number of Electrodes: | 8 | 8 | 6 | 18 | 18 | 23 |
| Fully Programmable: | X | X | X | X | X | ✓ |
| Fully Embedded: | ✓ | ✓ | X | X | ✓ | ✓ |

In contrast with the previously-presented solutions, exploiting an optimized near-threshold micro-architecture and the most advanced FD-SOI technology, the PULP platform allows achieving high performance and energy efficiency, coupled with the high versatility of programmable processors. Although the proposed implementation has been applied to a dataset with 23 electrodes due to the availability of the dataset, in the context of seizure detection applications, the availability of a fully-programmable platform allows trading the detection latency with a larger number of electrodes, addressing the key challenges highlighted by the trend in research, which goes toward the design of dense multichannel systems employing up to 128 electrodes [6–8]. Furthermore, system programmability is preferable to deal with the processing chains typical of most other biomedical applications, which need to be often updated or tuned during the life-time of a system [26]. In this work, we present the optimized implementation of a seizure-detection processing chain on PULP, consisting of a dimensionality reduction, feature extraction and classification steps [7,27]. We demonstrate the flexibility and the scalability of the PULP platform, able to detect a seizure on 23 EEG channels in 5 ms, improving performance by 140× with respect to state of the art MCUs. Moreover, exploiting the capabilities of the PULP platform allows satisfying the real-time and power consumption constraints through a fully-embedded programmable SoC, without the need for involving specific hardware accelerators or external computational devices.

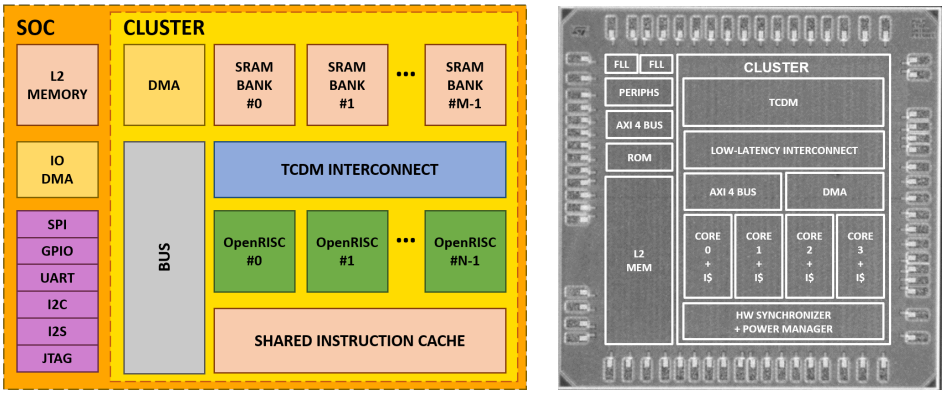
The application presented in this work is first implemented in MATLAB to verify functionality and validate the processing chain. Then, a C code version is implemented on the PULP platform to test the performance on a ULP embedded platform. The first version of this framework is based on floating point math and was presented in [28]. The evolution proposed in this work targets the optimization of the computational efforts (i.e., power consumption and latency), exploiting fixed-point operations. However, this reduced precision approach might be detrimental for algorithms requiring high precision and dynamic range and for those requiring iterative methods [16]. In a parallel processing platform, this trade-off might also be affected by the parallelism of the architecture. Thanks to the flexibility given by the software programmability of PULP, we show that for each configuration of the platform (e.g., 1, 2, 4, 8 cores) it is possible to choose the best trade-off between precision and computation time, optimizing the energy consumption for each kernel of the application, reducing the execution

energy by 1.4× with respect to a pure floating-point implementation and by 1.5× with respect to a pure fixed-point implementation.

## 3. PULP Platform

### 3.1. Cluster Architecture

This section describes the PULP platform, focusing on the architecture of its third embodiment fabricated in 28-nm UTBB FD-SOI technology [1]. The computational engine of the SoC is a cluster with a parametric number (2–16) of cores (Figure 1a). The processors are based on power-optimized four-pipeline stage micro-architecture implementing the OpenRISC ISA [29], featuring full forwarding with single stalls only on load-use and mispredicted branches. The original OpenRISC ISA and the micro-architecture of the core have been enhanced for energy-efficient digital signal processing, supporting zero-overhead hardware loops, L0 buffer, load and store operations embedding pointer arithmetic and power management instructions. The platform supports optional integration of floating-point units [30], suitable to deal with applications requiring high precision and dynamic range. The cluster features a shared instruction cache with L0 buffer and support for instruction broadcasting that greatly reduces the pressure on the cache banks with respect to a private solution, resulting in a much higher energy efficiency [31]. The cluster relies on an L1 explicitly-managed multi-banked Tightly-Coupled Data Memory (TCDM) for data access, avoiding memory coherency overhead of data caches and greatly increasing area and energy efficiency. Each logical bank can be implemented as a heterogeneous memory, either composed of SRAMs or latch-based Standard Cell Memory (SCM) banks. Instruction caches can also be implemented with SCMs. The usage of SCMs for the implementation of frequently-accessed memory banks significantly improves energy efficiency, since energy/access of SCM is significantly lower than that of SRAMs for the relatively small cuts needed in L1 instruction and data memories [32]. Depending on the availability of low-voltage memories in the targeted implementation technology, different ratios of SCM and SRAM memory can be instantiated at design time.




(**a**) (**b**)

**Figure 1.** A general view of the Parallel Ultra-Low Power (PULP) architecture (**a**) and the layout of the PULPv3 chip used for performance and power characterization (**b**).

### 3.2. SoC Architecture

Off-cluster (L2) memory and peripheral accesses are managed by a low power Direct Memory Access (DMA) tightly coupled to the TCDM featuring an ultra-low-latency programming interface (just 10 cycles for transfer configuration), up to 16 outstanding transactions and multiple physical channels [33]. At the SoC level, several peripherals are available including two SPI interfaces (one master and one slave), I2C, I2S, a camera interface, GPIOs, a boot-up ROM and a Joint Test Action Group (JTAG) interface for debugging purposes. The SPI interfaces can be configured in single mode or quad mode, useful to interface high-bandwidth components such as off-chip SRAMs or FLASH

memories. The SPI slave can be configured as a master, and a set of enable signals placed on both SPI interfaces allow the SoC to interface to up to four slave peripherals, such as eight-channel ADC suitable for EEG signal acquisition. To reduce the overall number of pads and make a low-cost wire bonding packaging of the SoC suitable for pervasive applications, the IO interfaces are multiplexed. The SoC supports up to four configurations of the IO matrix, programmed through a memory-mapped interface accessible by the cores. Thanks to the presented peripheral architecture, the SoC is then capable of operating either in stand-alone mode or as an accelerator of a standard host microcontroller (e.g., an ARM Cortex M processor), which provides programming legacy and offloads performance-critical parallel tasks to PULP [34]. In order to operate at the best energy point across a wide range of workloads, the PULP cluster and the rest of the SoC are in different power and clock domains. Fine-grained tuning of the cluster and SoC frequencies is managed by two FLLs (Frequency-Locked Loops).

### *3.3. Programming Model and Toolchain*

The PULP platform relies on OpenMP 3.0 parallel library that operates on top of the GCC 4.9 programming toolchain. The OpenMP implementation is based on a highly-optimized bare-metal library [35], which avoids the presence of an operating system that would introduce huge software overheads, not suitable for ultra-low-power parallel accelerators. To achieve high energy efficiency, PULP includes special hardware for accelerating key software patterns, such as barriers. Moreover, to reduce the power wasted by unused cores when worker threads are idling (e.g., in sequential regions of the program), PULP supports a clock-gating-based thread docking scheme to reduce the power of idle cores. The control of power management knobs is fully integrated in the OpenMP runtime, hence completely transparent to the programmer. The PULP platform features a set of software tools including a virtual platform and the support for parallel profiling, to implement, debug and profile applications that run on the architecture. The toolchain was used in this work to evaluate the parameters of the system, simulating architectural configurations not necessarily implemented on the silicon prototypes, such as the number of processors and the presence of floating-point units.

## 4. Seizure Detection on the PULP Architecture

### *4.1. Seizure Detection Algorithm*

This section describes the implementation of the seizure detection application on the PULP platform. The application was chosen as a benchmark for PULP as representative for a wide range of bio-processing applications, which consist of the three following steps: dimensionality reduction, feature extraction and classification [7,27]. A block diagram of the application is shown in Figure 2.

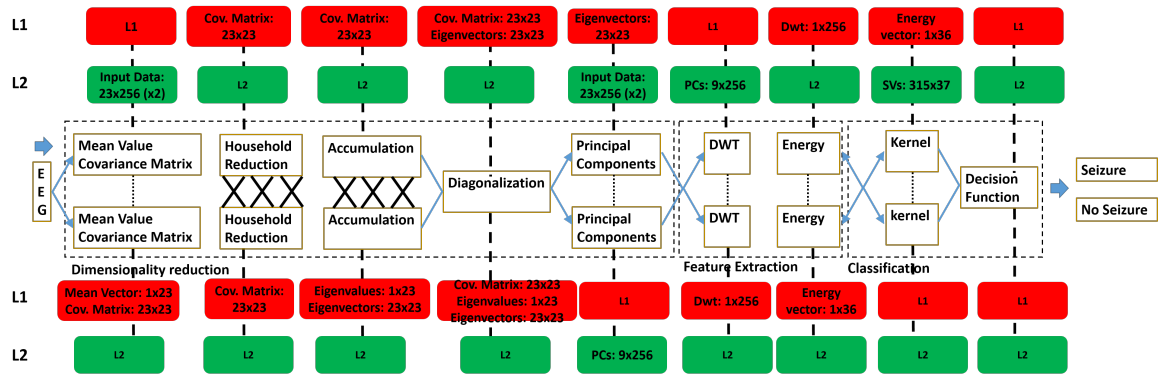



**Figure 2.** Seizure detection computational kernels.

Principal Component Analysis (PCA) is a commonly-used algorithm in the processing of EEG signals [36]. Through an orthogonal transformation, possibly correlated variables ($p$ acquisition

channels) are transformed into a set of linearly uncorrelated components (l < p). The input matrix containing the samples acquired from the EEG sensors is converted into a new coordinates system by the linear transformation:

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{P} \tag{1}$$

where $\mathbf{X}_{n\times p}$, $\mathbf{P}_{p\times l}$ and $\mathbf{Y}_{n\times l}$ are respectively input, transformation and reduced matrices. The number of Principal Components (PCs), which form the reduced matrix, is selected based on the variance retained by the PCs. In this application, the number of variables is reduced from 23 to nine, preserving more than 90% of the variance contained in the input data. Dimensionality reduction has the advantages of reducing memory requirements and decreasing the computational effort of the following processing steps of the application, providing a faster response and a higher energy efficiency.

After dimensionality reduction, the feature extraction step is used to extract significant information from the PCs. Features can be extracted in the time domain, in the frequency domain or in the time-frequency domain. Among the time-frequency domain solutions, Discrete Wavelet Transform (DWT) is widely used for feature extraction in several biosignal processing applications [37]. Through a bank of Low Pass (LPF) and High Pass Filters (HPF), the signal is decomposed at different levels of frequency resolution. The results, up to a given level of frequency resolution $n$, are a series of coefficients in the frequency domain, called approximation coefficients ($a_{(n)}$) for the LPF and detail coefficients ($d_{(n)}$) for the HPF. Therefore, energy is computed from detail coefficients at each level of resolution, creating the energy vector (E).

$$\mathbf{E}_{d_{(n)}} = \sum_{i=0} \left| d_{(n)}[i]) \right|^2 \tag{2}$$

The energy coefficients are used as input for the pattern recognition stage. Machine learning provides several algorithms for this purpose; Support Vector Machine (SVM) is an algorithm contained in the framework of the statistical learning theory, which demonstrates better robustness and efficiency with respect to other solutions [38]. In this application, as data can belong to two different classes, a binary classifier is adopted. Starting from labeled data, solving a convex optimization problem allows maximizing the hyperplane between the two possible classes. Support Vectors (SVs) delimit the decision boundary, and they are used to classify new data. A kernel function is used to map data in a higher dimensional space. This transformation is mandatory when the decision boundary is highly non-linear. The classifier, taking as an input the SVM model, computes the distance of the new instance from the decision boundary. Thus, new instances are classified through the formula:

$$f(\mathbf{x}) = \sum_{i=1}^{N_{SV}} y_i \alpha_i K\langle \mathbf{x}, \mathbf{s}_i \rangle - \rho \quad \begin{cases} f(\mathbf{x}) > 0, \mathbf{x} \in Cl_1 \\ f(\mathbf{x}) < 0, \mathbf{x} \in Cl_2 \end{cases} \tag{3}$$

where $Cl_1$ and $Cl_2$ are the two possible classes; $\mathbf{x}$ and $\mathbf{s}_i$ are respectively input features and SVs; $\alpha_i$ $y_i$ are precalculated coefficients, $\rho$ is a bias term and $K\langle \cdot, \cdot \rangle$ represents the Radial Basis Function (RBF) kernel function:

$$K\langle \mathbf{x}, \mathbf{s}_i \rangle = \exp\left( -\frac{||\mathbf{x} - \mathbf{s}_i||}{2\sigma^2} \right) \tag{4}$$

For the training phase of the algorithm, 30% of the dataset is used to compute the model. The other samples are used to test the model and validate it. The parameters of the classifier are tuned calculating an ROC curve [39], obtained varying the C parameter (i.e., the parameter used to control the trade-off between miss-classifications and margin maximization). The separation hyperplane presents smaller and larger margins respectively for small and high C values. Sensitivity and specificity are computed as:

$$Sensitivity = \frac{TP}{TP + FN} \tag{5}$$

$$Specificity = \frac{TN}{FP + TN} \tag{6}$$

where FN, TP, FP and TN are respectively False Negative, True Positive, False Positive and True Negative. In Figure 3, the ROC curve is shown. According to the figure, the best value obtained is $C = 0.1$, which leads to a classification accuracy around 99%.

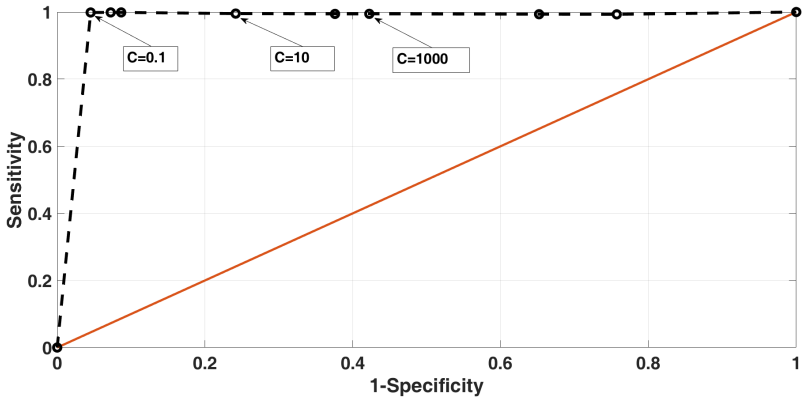



**Figure 3.** ROC curve for SVM performance evaluation.

### *4.2. Parallel Implementation on PULP*

This section describes the implementation of the seizure detection algorithm on the PULP platform. The processing chain is divided into three main stages: dimensionality reduction, feature extraction and pattern recognition. The first stage can be decomposed into five sub-kernels where each sub-kernel can be analyzed separately. The seizure detection algorithm is parallelized using OpenMP directives, scaling the execution of the application through 2, 4 up to 8 cores. In Figure 2, a block diagram describes the processing chain with particular emphasis on parallelization schemes and memory requirements.

#### 4.2.1. Dimensionality Reduction

PCA is the most challenging kernel of the processing chain. It is divided into five sub-kernels where Singular Value Decomposition (SVD) is performed by three of these kernels. SVD is a well-known algorithm that computes eigenvalues and eigenvectors of a matrix, the covariance matrix in our case. The method used to implement SVD is based on a common approach; it consists of transforming the starting matrix into a bi-diagonal form through successive Householder matrices and then, with an iterative method, diagonalizes the matrix through Givens matrices [40]. The output of this procedure is the eigenvectors' matrix, used to find the Principal Components (PCs). Due to the presence of dependencies through successive iterations, Householder reduction and accumulation kernels feature a complex parallelization scheme. In fact, the speed-ups are lower than the ideal. Moreover, parallel computation of these kernels is performed on small data chunks, and several barriers are necessary to synchronize all of the cores in different points of the execution, increasing the cost of OpenMP runtime. As already mentioned, diagonalization is an iterative method also featuring poor inclination to parallelization, showing the lowest speed-up among all of the kernels of the application. A large part of this kernel has to be executed sequentially, since it is affected by the pathological Amdahl bottleneck due to dependencies between iterations. Furthermore, the overhead of the OpenMP runtime increases since several synchronization points among cores are required. The first and the last kernels are mean value + covariance matrix and principal components. The first kernel computes and subtracts the mean values from the samples of each channel (i.e., to make them zero mean) and computes the $23 \times 23$ covariance matrix used to find the eigenvectors through the SVD procedure. The last one computes the reduced matrix that contains the PCs. These two kernels show high inclination for scaling among multiple cores achieving nearly ideal speed-ups.

The input matrix has dimension 23 × 256; thus, 23 kB of memory are allocated in L2. Through the DMA, data are transferred from L2 to L1 memory by a double buffering policy. Then, after subtracting the mean values from all samples, the covariance matrix is computed and stored in the L1 memory. At this point, SVD's kernels produce an eigenvector matrix of dimension 23 × 23 (i.e., around 2 kB). This matrix, multiplied by the input matrix, produces the reduced matrix of dimension 9 × 256 (9 kB) that is stored in L2 memory.

#### 4.2.2. Feature Extraction

After dimensionality reduction, the processing chain continues the execution performing the feature extraction. DWT is computed starting from the reduced matrix of dimension 9 × 256, and the energy contained in the signal is calculated up to the chosen resolution (i.e., four levels) and stored in L1 memory in the energy vector of dimension 1 × 36 (144 bytes). Each principal component can be processed separately because there are no dependencies between iterations. Hence, each OpenMP thread can operate on different vectors. As the number of components is not a multiple of the number of cores, the workload results in being unbalanced, limiting the speed-up of the parallel execution.

#### 4.2.3. Pattern Recognition

The next step consists of the SVM classifier. The energy vector computed in the previous step is passed as input to the SVM predict function that has the aim of predicting the class of this feature vector (i.e., seizure, no seizure). The prediction is done through a model stored in L2 memory. The model is composed of 315 SVs, each one of dimension 1 × 37 (i.e., 36 features and one coefficient). To store the model in L2, 46 kB are required. Data are transferred from L2 to L1 memory exploiting the double buffering policy using the DMA. In the multi-core version, each OpenMP thread can compute the kernel function separately on one SV. Due to the limited computation effort, the decision function is executed sequentially.

### *4.3. Fixed-Point Implementation*

To reach a high level of accuracy during the execution of the seizure detection application, double or single precision floating-point representation of data is often used in desktop processors, using EEG feature extraction and interpretation toolboxes for MATLAB such as EEGLAB [41]. However, processing floating-point data requires a high computational effort, especially in deeply-embedded processors where power consumption and energy are the main fundamental aspects that have to be taken into account during the design of the application. Floating-point operations can be either emulated via software or can be executed with dedicated hardware. The majority of floating-point processors represent real numbers following the IEEE 754 data format [42], using finite precision arithmetic. For a 32-bit word length, numbers are represented using one-bit for the sign, eight bits for the exponent and 23 bits for the mantissa. To improve energy efficiency, numbers can be represented in fixed-point format: unsigned, signed or two's compliment. A common representation for fixed-point number is the Qm.n format, where m and n are the number of bits used respectively for the Integer Word Length (IWL) and the Fractional Word Length (FWL) part of the number [43].

The most challenging part of the fixed point implementation is the PCA. This algorithm is characterized by a high dynamic range, where data move from small range to high range during the computational steps, which can lead to numeric overflows when fixed-point representation is used. For this kernel, a Q13.19 format is chosen (i.e., 13 bits for sign and IWL and 19 bit for FWL). This choice has been taken after an accurate analysis of the dynamics of the variables during the execution of the processing chain. Conversion from floating- to fixed-point representation is performed as:

$$((\mathbf{int})(\mathbf{x} * (\mathbf{int})(\mathbf{1} << \mathbf{FWL}))) \tag{7}$$

where x is a floating-point number, while the opposite operation that converts a fixed-point number to a floating-point is:

$$((\mathbf{float})\mathbf{y}/(\mathbf{float})(\mathbf{1} << \mathbf{FWL})) \tag{8}$$

where y is a fixed point number. Sums and subtractions are performed as a common integer operation; the result of a sum or a subtraction between two Q13.19 numbers is a Q13.19 number, while multiplications and divisions are more challenging. In fact, to avoid numeric overflows, a common practice is to cast the 32-bit operands to a 64-bit format [43]. The impact on the performance of this approach is considerable, since it implies a multiplication/division between 64-bit variables. To avoid performance degradation, two additional functions have been implemented to perform multiplications without casting variables to 64-bit format:

$$((\mathbf{A} >> \mathbf{9}) * (\mathbf{B} >> \mathbf{9}) >> \mathbf{1}) \tag{9}$$

$$((\mathbf{A} >> \mathbf{7}) * (\mathbf{B} >> \mathbf{7}) >> \mathbf{5}) \tag{10}$$

The first one (fixed_mulHR) provides a way to perform multiplications increasing the integer part of the number, avoiding overflows. On the contrary, the number of bits dedicated to the fractional part decreases, causing a loss of accuracy. The second one (fixed_mulHP) is used to perform multiplication through a lower range for the integer part with respect to the previous, providing a higher level of accuracy. After an accurate analysis of the dynamics of the variables (i.e., maximum and minimum values assumed by the variables), we have selected the functions to be instantiated in the different kernels of the PCA. Moreover, from this analysis, it is also possible to understand how many bits have to be shifted to avoid numeric overflows or to maintain as much accuracy as possible. In the diagonalize kernel, 32-bit multiplications can cause frequent overflows leading to the wrong results due to the high dynamic variation of the variables. Thus, for this kernel, 64-bit multiplications were chosen to avoid overflows, despite the higher computation time.

Regarding the classification stage, a common practice is to scale data before training the algorithm to create the model and the features to classify [44]. Scaling data provides several advantages. The first advantage regards the numeric ranges of the data: if the variation of the numeric ranges is high, then the greater can dominate among the smaller. The second advantage is related to calculations that become easier reducing the numeric ranges, simplifying the training phase [44]. For the nature of the EEG signal and the related energy values, a scaling factor between $10^{-5}$ and $10^{-7}$ is required. In the floating-point version of the application on PULP, we scale data after the energy kernel. Since the ranges of the input data can vary (i.e., input data that correspond to Class 1 can be up to two orders higher than the ones of Class 0), there is high risk of incurring numeric overflows in the fixed-point version. To avoid this problem, data are scaled before PCA kernel, and the scaling factor changes according to the range of the input data. The mean of the first entries of each channel that compose the input matrix is computed. If the value of the mean is above a given threshold, then the scaling factor is two orders higher than the one used for samples whose mean value is under the threshold. After the energy kernel, the feature vector is re-scaled to obtain the right order for the classification stage.

## 5. Experimental Results

### 5.1. Experimental Setup

An on-line dataset called CHB-MIT (Children's Hospital Boston-Massachusetts Institute of Technology) is used to take EEG data used in the simulation [45]. It contains samples acquired from 24 pediatric subjects suffering for intractable seizures acquired at 16-bit resolution with a sampling frequency of 256 Hz. EEG data during regular (no seizure) and abnormal (seizure) brain activity are taken from four patients randomly chosen within the dataset. To test the processing chain, we

have initially implemented and executed the application on MATLAB, verifying the accuracy gained from seizure recognition. Models are created using 30% of data (half no seizure, half seizure) of the subject and tested with the remaining 70%. The accuracy reached by the algorithm is around 92–99%. After validation, both the fixed-point and floating-point versions of the the processing chain were implemented and evaluated on the PULP platform. Both the fixed-point and floating-point versions are parallelized among multiple cores (2, 4 and 8 cores) to evaluate the scaling capabilities of the system and the related performance. To obtain reliable results as concerns classification accuracy, an implementation using 30,256 $\times$ 23 samples windows randomly chosen from the testing set belonging to the two classes (i.e., 70% Class 0, 30% Class 1) was simulated. The virtual platform of PULP was used to estimate the execution time and the energy consumption of the application. To characterize the power numbers of the PULP architecture, data were extracted from measurements on the silicon prototype of the PULPv3 SoC, shown in Figure 1b, fitted with post-layout simulations to characterize the breakdown of the power consumption among the components of the cluster and adapted to the configurations employed in the exploration by annotating the energy numbers on the virtual platform.

### *5.2. Seizure Detection Accuracy*

Table 2 shows the percentages of precision and accuracy derived from the execution of the floating-point and fixed-point processing chains on PULP. Precision is evaluated comparing the energy matrices obtained at the output of the DWT + ENERGYkernel, while accuracy is evaluated at the output of the SVM classifier. On the MATLAB implementation, used as a reference golden model, a 64-bit double precision format is employed, while PULP employs 32-bit single precision format. It is possible to note that comparing the energy matrix obtained at the output DWT + ENERGY kernel inn MATLAB and the PULP platform floating-point, the loss of precision is negligible (i.e., less than 0.1%). On the other hand, the fixed-point conversion causes a slight loss of precision due to rounding and saturation during the execution of the processing chain. The results obtained from the execution of the fixed-point application lead to an average loss of precision of 6.5% compared to the floating point version and to an average accuracy of 92% for the floating-point application and 89% from the fixed-point version, which is aligned with the state of the art [46–48].

**Table 2.** Accuracy and Precision obtained from MATLAB and PULP platform floating- and fixed-point executions.

| | **MATLAB** | **FLOATING-POINT** | | **FIXED-POINT** | |
|---|---|---|---|---|---|
| **Subjects** | Accuracy% [1] | Precision% [2] | Accuracy% [1] | Precision% [2] | Accuracy% [1] |
| **S01** | 93 | 100 | 93 | 91 | 80 |
| **S02** | 100 | 100 | 100 | 90 | 100 |
| **S03** | 74 | 100 | 74 | 97 | 74 |
| **S04** | 100 | 100 | 100 | 96 | 100 |
| **Mean** | 91.75 | 100 | 91.75 | 93.50 | 88.50 |

[1] Accuracy obtained from the classifier; [2] precision maintained after DWT + ENERGY kernel w.r.t. MATLAB results.

### *5.3. Evaluation of Execution Performance*

Tables 3 and 4 summarize the execution time (clock cycles) of the seizure detection application on the PULP platform, in its floating-point and fixed-pint embodiment, respectively. In the sequential implementation, PCA requires 80% of the overall computation time, while DWT, energy and SVM require the remaining computational load (20%). When the algorithm runs exploiting parallel processing over the multiple cores of PULP, the execution time reduces by up to 5$\times$ in the floating-point version and 3.44 in the fixed-point version. If we first analyze Table 3, showing the results of the floating-point implementation, we can notice that the kernels with higher parallelism, such as mean value + covariance, compute PC and SVM, accounting for more than 75% of the overall computational load, feature nearly ideal speed-ups. On the other hand, other kernels such as Householder reduction

and accumulate require frequent parallel computations on small data chunks interleaved with synchronization points, which increases the overhead of the OpenMP runtime and degrades the parallel performance. Finally, the DWT + energy kernel is affected by workload unbalance, since it requires the elaboration of nine principal components on eight processors, limiting the speed-up of this kernel to 4.38×. Diagonalize is an iterative kernel affected by the pathological Amdahl bottleneck caused by the dependencies between matrix elements calculated during the iterations, which force most of this kernel to be executed sequentially, hence leading to a very poor parallel speed-up (i.e., 2× on eight cores). The situation is even worse in the fixed-point implementation due to the frequent scaling (Section 4.3) and the difficulty to converge in the diagonalize kernel due to loss of precision.

**Table 3.** Execution of floating-point seizure detection on the embedded computing platform.

| | PULP 1 Core | | PULP 2 Cores | | | PULP 4 Cores | | | PULP 8 Cores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | kCycles | load% | kCycles | Speed-up [a] | E.Save% [b] | kCycles | Speed-up [a] | E.Save% [b] | kCycles | Speed-up [a] | E.Save% [b] |
| **PCA** | 1902 | 79.25 | 1057 | 1.80 | | 611 | 3.11 | | 374 | 5.08 | |
| Mean + Covariance | 1018 | 53.50 | 514 | 1.98 | 0 | 284 | 3.59 | 0 | 146 | 6.95 | 0.60 |
| Householder Reduction | 151 | 8.00 | 85 | 1.78 | 4.00 | 52 | 2.92 | 11.45 | 35 | 4.32 | 20.30 |
| Accumulate | 90 | 4.70 | 50 | 1.81 | 2.50 | 29 | 3.05 | 7.45 | 19 | 4.60 | 14.80 |
| Diagonalize | 228 | 12.00 | 162 | 1.38 | 11.00 | 132 | 1.73 | 22.85 | 116 | 1.97 | 32.00 |
| Compute PC | 413 | 21.80 | 207 | 1.99 | 0 | 105 | 3.93 | 0 | 58 | 7.12 | 0 |
| **DWT + ENERGY** | 169 | 7.05 | 94 | 1.80 | 5.00 | 57 | 2.98 | 14.30 | 39 | 4.38 | 26.50 |
| **SVM** | 328 | 13.70 | 175 | 1.87 | 0 | 98 | 3.33 | 0 | 60 | 5.43 | 0.65 |
| **TOTAL** | 2400 | 100 | 1326 | 1.86 | 4.50 | 766 | 3.13 | 12.00 | 475 | 5.05 | 21.00 |

[a] Speed-up with respect to single-core PULP platform; [b] Energy Saving with power management techniques.

**Table 4.** Execution of fixed-point seizure detection on embedded computing platform.

| | PULP 1 Core | | PULP 2 Cores | | | PULP 4 Cores | | | PULP 8 Cores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | kCycles | load% | kCycles | Speed-up [a] | E.Save% [b] | kCycles | Speed-up [a] | E.Save% [b] | kCycles | Speed-up [a] | E.Save% [b] |
| **PCA** | 2159 | 83.25 | 1392 | 1.55 | | 923 | 2.34 | | 681 | 3.17 | |
| Mean + Covariance | 995 | 42.2 | 536 | 1.86 | 0 | 284 | 3.50 | 0.40 | 143 | 6.97 | 0.60 |
| Householder Reduction | 246 | 13.00 | 170 | 1.45 | 4.25 | 125 | 1.97 | 14.40 | 106 | 2.33 | 28.60 |
| Accumulate | 109 | 6.60 | 85 | 1.28 | 1.50 | 50 | 2.18 | 6.80 | 33 | 3.30 | 17.70 |
| Diagonalize | 405 | 20.60 | 395 | 1.03 | 9.75 | 355 | 1.14 | 25.20 | 337 | 1.20 | 44.60 |
| Compute PC | 397 | 17.60 | 204 | 1.95 | 0 | 104 | 3.82 | 0 | 58 | 6.84 | 0 |
| **DWT + ENERGY** | 136 | 4.78 | 76 | 1.79 | 2.50 | 46 | 2.97 | 10.00 | 31 | 4.38 | 25.15 |
| **SVM** | 304 | 11.97 | 164 | 1.86 | 0 | 85 | 3.58 | 0 | 45 | 6.75 | 1.15 |
| **TOTAL** | 2599 | 100 | 1631 | 1.60 | 3.00 | 1053 | 2.47 | 11.20 | 756 | 3.44 | 26.30 |

[a] Speed-up with respect to single-core PULP platform; [b] Energy Saving with power management techniques.

Despite that the poor parallelization affinity of some of the kernels of the processing chain degrades performance with respect to the ideal case (i.e., 8× speed-up when executing on eight cores), preventing further voltage and frequency scaling for a given real-time requirement, the PULP platform tackles the problem through fine-grained clock gating of the idle processors of the cluster, as described in Section 3.3. This concept is highlighted in both Tables 3 and 4, in the columns showing the energy savings achieved by activating the fine-grained power management techniques employed in the PULP platform. As such, kernels featuring almost ideal speed-ups do not show any advantage since in these kernels, all of the resources of the cluster are (almost) fully utilized. On the other hand, the energy saving of heavily parallelizable kernels, such as Diagonalize, can be up to 45%.

### *5.4. Hybrid Implementation Performance*

Table 5 shows a comparison between the number of cycles derived from the fixed- and floating-point processing chain on the PULP platform. It is noteworthy that if we consider execution time (i.e., number of cycles) as a comparison metric, the floating-point implementation provides better performance than the fixed-point implementation, even if floating-point operations require multiple cycles (i.e., two cycles for floating-point additions and multiplications). This is due to the relevant overhead required to perform complex fixed-point operations, such as 64-bit multiplications and divisions, and due to the additional overhead required to perform the scaling of the samples,

which modifies the dynamic of the data at the boundary of some of the kernels. As shown in Table 5, Householder reduction and diagonalize kernels are the ones with the highest ratio between float and fixed execution time. Indeed, these kernels are the most computational dense, hence the overhead derived from function calls to perform multiplications, divisions and square roots significantly affects performance. Furthermore, the diagonalize kernel is computed with an iterative method that ends when the convergence's condition is reached. Thus, using fixed-point arithmetic implies a higher execution time needed to converge.

If we consider energy consumption as a comparison metric, the trade-off becomes more interesting. On the one hand, the floating-point algorithm provides faster execution, allowing a higher degree of voltage and frequency scaling for a given real-time constraint, which improves energy efficiency. On the other hand, the execution of the kernels on simpler integer units can reduce the power consumption of the cluster by up to 35% with respect to the floating-point execution, considering the same supply voltage for both implementations. This concept is well highlighted in Table 6, which compares the energy consumption of the kernels on the floating-point and fixed-point versions. Depending on the kernel, but also on the number of cores in the platform, the different trade-offs in terms of sequential execution time, cost of operations and parallelization cause differences, often relevant, on the best choice concerning energy.

**Table 5.** Execution time on the PULP platform.

| | **PULP 1 Core** | | | **PULP 2 Cores** | | | **PULP 4 Cores** | | | **PULP 8 Cores** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | kC.Fl [a] | kC.Fx [b] | R [c] | kC.Fl [a] | kC.Fx [b] | R [c] | kC.Fl [a] | kC.Fx [b] | R [c] | kC.Fl [a] | kC.Fx [b] | R [c] |
| **PCA** | | | | | | | | | | | | |
| Mean + Covariance | 1018 | **995** | 0.98 | **514** | 536 | 0.98 | 284 | 284 | 1.01 | 146 | **143** | 0.99 |
| Householder Reduction | **151** | 246 | 1.96 | **85** | 170 | 2.21 | **52** | 125 | 2.60 | **35** | 106 | 3.17 |
| Accumulate | **90** | 109 | 1.74 | **50** | 85 | 1.72 | **29** | 50 | 1.71 | **19** | 33 | 1.69 |
| Diagonalize | **228** | 405 | 2.10 | **162** | 395 | 2.46 | **132** | 355 | 2.78 | **116** | 337 | 2.98 |
| Compute PC | 413 | **397** | 0.99 | 207 | **204** | 0.99 | 105 | **104** | 0.98 | 58 | 58 | 1.00 |
| **DWT + ENERGY** | 169 | **136** | 0.80 | **94** | 76 | 0.81 | 57 | **46** | 0.81 | 39 | **31** | 0.79 |
| **SVM** | 328 | **304** | 1.04 | 175 | **164** | 0.99 | 98 | **85** | 0.91 | 60 | **45** | 0.79 |
| **TOTAL** | 2400 | 2599 | 1.19 | 1326 | 1631 | 1.29 | 766 | 1053 | 1.45 | 475 | 756 | 1.68 |

[a] Floating-point execution time in thousands of cycles; [b] Fixed-point execution time in thousands of cycles; [c] Ratio between float and fixed execution.

**Table 6.** Energy per frame on the PULP platform in µJ with 5-ms real-time constraint.

| | **PULP 1 Core** | | **PULP 2 Cores** | | **PULP 4 Cores** | | **PULP 8 Cores** | |
|---|---|---|---|---|---|---|---|---|
| **Kernel** | E.Fl. (µJ) [a] | E.Fx. (µJ) [b] | E.Fl. (µJ) [a] | E.Fx. (µJ) [b] | E.Fl. (µJ) [a] | E.Fx. (µJ) [b] | E.Fl. (µJ) [a] | E.Fx. (µJ) [b] |
| **PCA** | | | | | | | | |
| Mean + Covariance | **3206** | 4409 | 1495 | **1468** | 1100 | **789** | 886 | **471** |
| Householder Reduction | **475** | 1307 | **236** | 521 | **177** | 317 | **169** | 259 |
| Accumulate | **202** | 689 | **140** | 245 | **105** | 130 | 101 | **89** |
| Diagonalize | **720** | 2121 | **428** | 1068 | **394** | 759 | **480** | 629 |
| Compute PC | **1300** | 1850 | 601 | **596** | 407 | **286** | 353 | **190** |
| **DWT + ENERGY** | 533 | **191** | 259 | **117** | 188 | **81** | 173 | **63** |
| **SVM** | 1034 | **519** | 510 | **283** | 382 | **179** | 366 | **130** |
| **TOTAL** | 7576 | – | 3548 | 5201 | 2634 | 3295 | 2279 | 2351 |

[a] Energy per frame Floating-point; [b] Energy per frame Fixed-point.

To this end, as a further algorithmic optimization, a hybrid algorithm is proposed, on the bases of the energy per frame obtained on both floating- and fixed-point execution. The flexibility of the PULP platform allows one to vary the approach used for execution (to switch from float to fixed and vice versa) on the basis of the kernel and the configuration we are using, to reduce as much as possible the energy consumption. As shown in Figure 4, in this way, even if the execution time of the hybrid approach is nearly similar to that of the floating-point implementation, the energy per frame can

be reduced by up to 1.5× with respect to the fixed-point implementation and 1.4× with respect to the floating-point implementation for execution on eight cores, significantly improving the energy efficiency of the application.

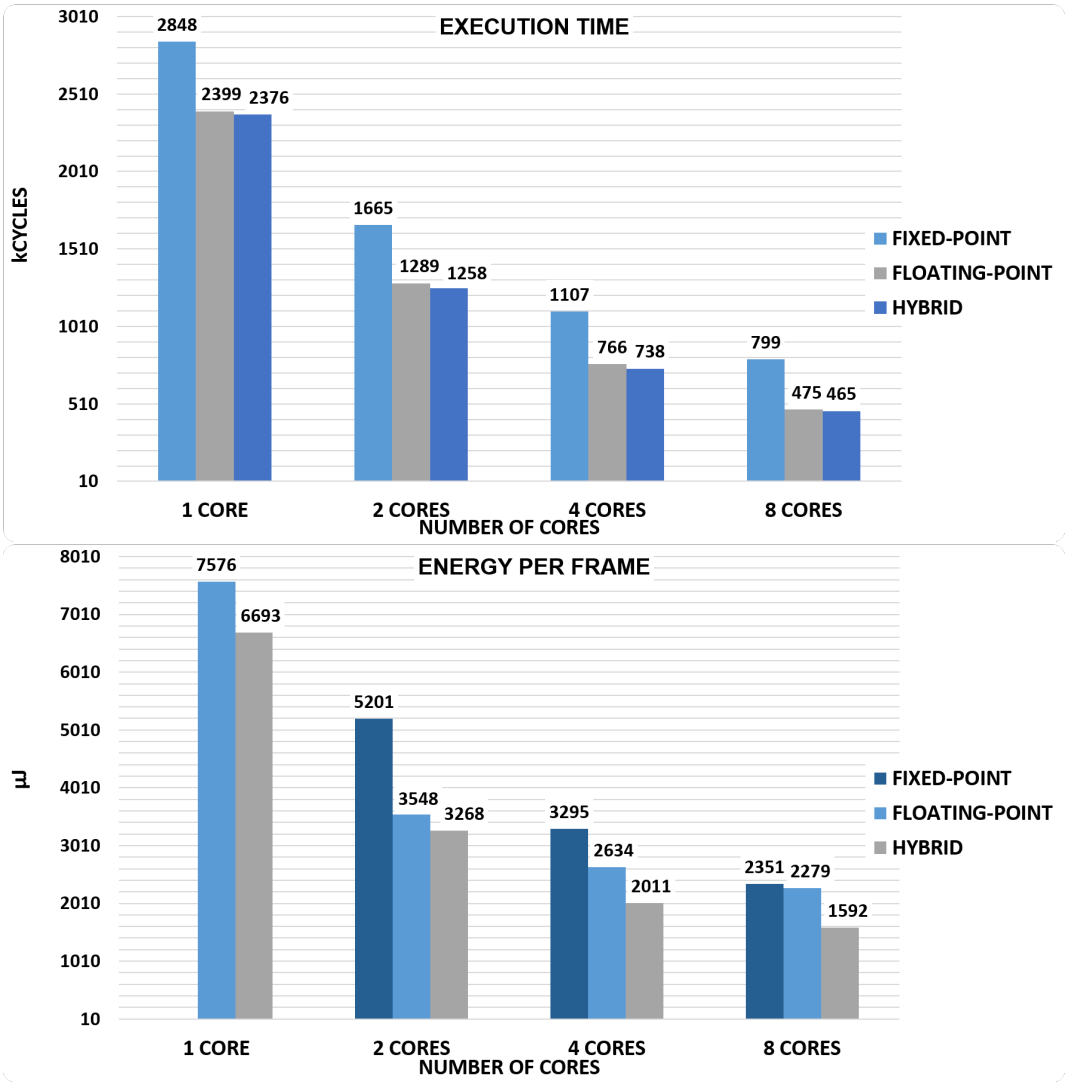



**Figure 4.** Comparison between floating-, fixed-point and hybrid execution time and energy per frame.

### *5.5. Energy Considerations and Comparison with Commercial MCUs*

The seizure detection processing chain was implemented and executed on two commercial MCUs integrating an ARM Cortex M4 processor and hardware Floating Point Unit (FPU); Ambiq Apollo and STM32F427, and on the PULP platform with a configuration with a single core and with multiple cores (2, 4, 8 cores). The power consumptions of Ambiq Apollo and STM32F427 processors are 115 µW/MHz and 600 µW/MHz, respectively, extracted from the datasheets of the two MCUs. The processing chain was tested imposing three real-time latency constraints: 500 ms, 50 ms and 5 ms. The 5-ms constraint is very restrictive and not required for medical purposes, but it is useful to evaluate the performance of the system. Operating frequencies and power consumption are calculated at each real-time constraint, taking into account the time for which only the core master is active and the other cores are idle and can be put in sleep mode. Operating frequency refers to the execution of the application on a data frame composed by 256 samples acquired from 23 channels, within the given detection latency.

Figure 5 shows the results in terms of operating frequency required to satisfy the seizure detection latency constraints and the related values of energy per frame on the PULP platform and other common

MCUs widely used in embedded systems. Ambiq Apollo can satisfy only the 500-ms constraint, as it can achieve a maximum operating frequency equal to 24 MHz. STM32F427 can operate to a maximum frequency of 180 MHz; it satisfies the latency constraint of 50 ms, but it is not sufficient for the 5-ms detection latency. PULP is able to satisfy all of the detection latency constraints except for the single-core fixed-point execution, which requires an operating frequency equal to 570 MHz for the 5-ms constraint (maximum operating frequency of 500 MHz). In the hybrid implementation, the most energy efficient, it consumes an average power of 27 µW, 274 µW and 3.18 mW, respectively, for 500-ms, 50-ms and 5-ms detection latency. The operating frequency required to satisfy the detection latency constraints on the PULP platform decreases by increasing the number of cores. In this way, it is possible to reduce the supply voltage leading to an improvement in power density (quadratic dynamic power dependency with supply voltage).

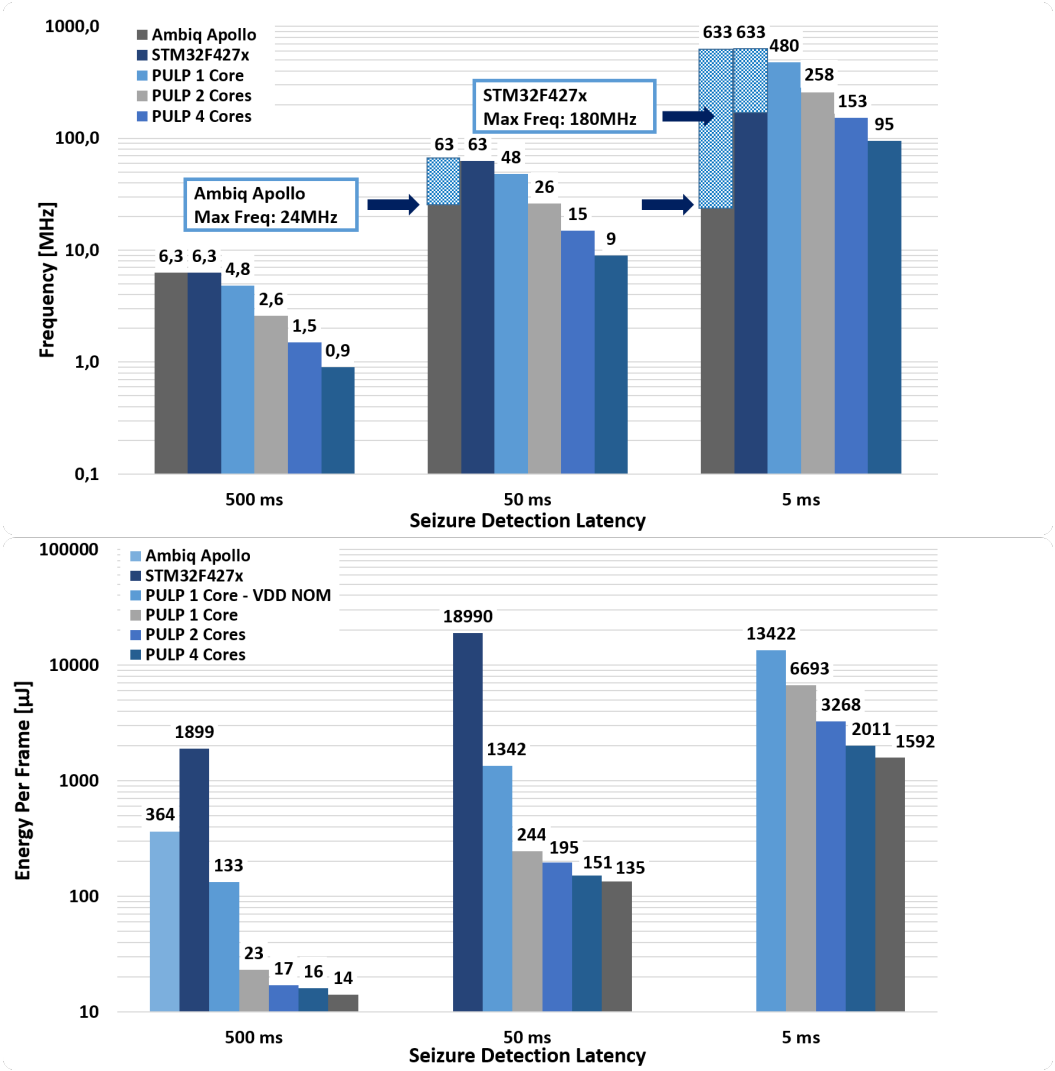



**Figure 5.** Operating frequency and energy per frame of the hybrid approach on the PULP and commercial MCUs.

Comparing the energy per frame of the commercial MCUs with the single-core PULP platform working at nominal supply voltage leads to 14×–3× lower energy. This is mainly due to the technological gap, as Ambiq Apollo and STM32F427 are both implemented with a 90-nm CMOS technology. In the single-core processing, a further improvement of 6× is achieved introducing near threshold computing and from 16× up to 83× compared to commercial MCUs. Introducing parallel programming leads to reducing the energy by 75%, 85% and 88%, respectively, with 2, 4 and 8 cores,

with respect to the single-core execution at nominal voltage. These results show the great advantages of using near-threshold computing and parallelism on a flexible and scalable architecture, leading to a scaling up of the complexity of the system and satisfying the detection latency requirements. All of these approaches allow one to drastically extend the battery life. This is an important aspect for embedded systems, such as implantable neurological devices, which have the aim of addressing a power budget compatible with implantable energy harvesters [49].

## 6. Conclusions

This work presented a seizure detection application on a Parallel Ultra-Low-Power (PULP) platform. Starting from the architecture and silicon characterization of the third embodiment of the PULP platform, implemented in 28-nm FD-SOI technology, we analyze the optimized implementation of the seizure detection algorithm, analyzing different solutions, which explore the performance/accuracy trade-off. The results of our exploration show that PULP platform is an appropriate solution for a biomedical application like seizure detection, and its flexibility and efficient micro-architecture lead to lower execution times and power consumption with respect to other commercial solutions (up to 140× lower energy). Moreover, the effect of the conversion from floating- to fixed-point format is shown in terms of the precision in the results and accuracy in the classification during a real-time simulation using EEG data acquired from four epileptic subjects. Furthermore, after an accurate analysis of the performance of each kernel in each configuration (i.e., single or multiple-cores), a hybrid algorithm was implemented combining floating- and fixed-point approaches, obtaining a further energy gain of 39% and 30% with respect to fixed- and floating-point version. In conclusion, it was shown that PULP is able to satisfy the computational requirements of a complex biomedical application for seizure detection without exceeding the power envelope of a few mW, from 13.4 mW with a single core to 3.18 mW with eight cores, imposing a seizure detection latency equal to 5 ms. Comparing the execution of the algorithm on the PULP platform with eight cores with respect to commercial MCUs, we obtain a 26× energy gain (more then 96% less) with respect to Ambiq Apollo and up to 140× compared to STM32F427x (99.30% energy savings).

**Acknowledgments:** This work has been partially supported by the FP7 ERC Advanced project MULTITHERMAN (Multiscale Thermal Management of Computing Systems) Grant Agreement No. 291125 and by the OPRECOMP (Open transPREcision COMPuting) project founded from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 732631.

**Author Contributions:** Fabio Montagna and Simone Benatti conceived of and designed the algorithmic implementation. Davide Rossi conceived of and designed the embedded implementation and power models. Fabio Montagna performed the experiments. All of the authors contributed equally in writing the paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Rossi, D. Sub-pJ per operation scalable computing: The PULP experience. In Proceedings of the 2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), Burlingame, CA, USA, 10–13 October 2016; pp. 1–3.
2. United Nations. Available online: http://www.un.org/ (accessed on 1 December 2016)
3. Al-Otaibi, F.A.; Hamani, C.; Lozano, A.M. Neuromodulation in epilepsy. *Neurosurgery* **2011**, *69*, 957–979.
4. Nune, G.; DeGiorgio, C.; Heck, C. Neuromodulation in the treatment of epilepsy. *Curr. Treat. Options Neurol.* **2015**, *17*, 43.
5. Sun, F.T.; Morrell, M.J. Closed-loop neurostimulation: The clinical experience. *Neurotherapeutics* **2014**, *11*, 553–563.
6. Acharya, U.R.; Sree, S.V.; Alvin, A.P.C.; Suri, J.S. Use of principal component analysis for automatic classification of epileptic EEG activities in wavelet framework. *Expert Syst. Appl.* **2012**, *39*, 9072–9078.
7. Subasi, A.; Gursoy, M.I. EEG signal classification using PCA, ICA, LDA and support vector machines. *Expert Syst. Appl.* **2010**, *37*, 8659–8666.
8. Wang, C.; Zou, J.; Zhang, J.; Wang, M.; Wang, R. Feature extraction and recognition of epileptiform activity in EEG by combining PCA with ApEn. *Cogn. Neurodyn.* **2010**, *4*, 233–240.

9. Neurospace. Available online: http://www.neuropace.com/ (accessed on 1 November 2015).
10. Medtronic. Available online: http://www.medtronic.com/ (accessed on 1 November 2015).
11. Yoo, J.; Yan, L.; El-Damak, D.; Altaf, M.B.; Shoeb, A.; Yoo, H.J.; Chandrakasan, A. An 8-channel scalable EEG acquisition SoC with fully integrated patient-specific seizure classification and recording processor. In Proceedings of the 2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco, CA, USA, 19–23 February 2012, pp. 292–294.
12. Lee, K.H.; Verma, N. A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *IEEE J. Solid State Circuits* **2013**, *48*, 1625–1637.
13. Benatti, S.; Milosevic, B.; Farella, E.; Benini, L. A Prosthetic Hand Body Area Controller Based on Efficient Pattern Recognition Control Strategies. *Sensors* **2017**, 17, 869.
14. Dreslinski, R.G.; Wieckowski, M.; Blaauw, D.; Sylvester, D.; Mudge, T. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proc. IEEE* **2010**, *98*, 253–266.
15. Rossi, D.; Pullini, A.; Loi, I.; Gautschi, M.; Gurkaynak, F.K.; Teman, A.; Constantin, J.; Burg, A.; Miro-Panades, I.; Beigné, E.; et al. 193 MOPS/mW @ 162 MOPS, 0.32 V to 1.15 V voltage range multi-core accelerator for energy efficient parallel and sequential digital processing. In Proceedings of the 2016 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS XIX), Yokohama, Japan, 20–22 April 2016; pp. 1–3.
16. Golub, G.H.; Van der Vorst, H.A. Eigenvalue computation in the 20th century. *J. Comput. Appl. Math.* **2000**, *123*, 35–65.
17. Ilyas, M.Z.; Saad, P.; Ahmad, M.I. A survey of analysis and classification of EEG signals for brain-computer interfaces. In Proceedings of the 2015 2nd IEEE International Conference onBiomedical Engineering (ICoBE), Penang, Malaysia, 30–31 March 2015; pp. 1–6.
18. Escabí, M.A.; Read, H.L.; Viventi, J.; Kim, D.H.; Higgins, N.C.; Storace, D.A.; Liu, A.S.; Gifford, A.M.; Burke, J.F.; Campisi, M.; et al. A high-density, high-channel count, multiplexed $\mu$ECoG array for auditory-cortex recordings. *J. Neurophysiol.* **2014**, *112*, 1566–1583.
19. Chen, W.M.; Chiueh, H.; Chen, T.J.; Ho, C.L.; Jeng, C.; Ker, M.D.; Lin, C.Y.; Huang, Y.C.; Chou, C.W.; Fan, T.Y.; et al. A fully integrated 8-channel closed-loop neural-prosthetic CMOS SoC for real-time epileptic seizure control. *IEEE J. Solid State Circuits* **2014**, *49*, 232–247.
20. Patel, K.; Chua, C.P.; Fau, S.; Bleakley, C.J. Low power real-time seizure detection for ambulatory EEG. In Proceedings of the 2009 3rd IEEE International Conference on Pervasive Computing Technologies for Healthcare, PervasiveHealth 2009, London, UK, 1–3 April 2009; pp. 1–7.
21. Verma, N.; Shoeb, A.; Bohorquez, J.; Dawson, J.; Guttag, J.; Chandrakasan, A.P. A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system. *IEEE J. Solid State Circuits* **2010**, *45*, 804–816.
22. Kwong, J.; Chandrakasan, A.P. An Energy-Efficient Biomedical Signal Processing Platform. *IEEE J. Solid State Circuits* **2011**, *46*, 1742–1753.
23. Kwong, J.; Ramadass, Y.; Verma, N.; Koesler, M.; Huber, K.; Moormann, H.; Chandrakasan, A. A 65 nm Sub-Vt Microcontroller with Integrated SRAM and Switched-Capacitor DC-DC Converter. *IEEE J. Solid State Circuits* **2009**, *44*, 115–126.
24. Volder, J.E. The CORDIC trigonometric computing technique. *IRE Trans. Electron. Comput.* **1959**, *EC-8*, 330–334.
25. Walther, J.S. A unified algorithm for elementary functions. In Proceedings of the Spring Joint Computer Conference, Atlantic City, NJ, USA, 18–20 May 1971; ACM: New York, NY, USA, 1971; pp. 379–385.
26. Causo, M.; Benatti, S.; Frappé, A.; Cathelin, A.; Farella, E.; Kaiser, A.; Benini, L.; Rabaey, J. Sampling Modulation: An Energy Efficient Novel Feature Extraction for Biosignal processing. In Proceedings of the 2016 IEEE Biomedical Circuits and Systems Conference (BioCAS), Shanghai, China, 17–19 October 2016.
27. Liu, Y.; Zhou, W.; Yuan, Q.; Chen, S. Automatic seizure detection using wavelet transform and SVM in long-term intracranial EEG. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2012**, *20*, 749–755.
28. Benatti, S.; Montagna, F.; Rossi, D.; Benini, L. Scalable EEG seizure detection on an ultra low power multi-core architecture. In Proceedings of the 2016 IEEE Biomedical Circuits and Systems Conference (BioCAS), Shanghai, China, 17–19 October 2016; pp. 86–89.
29. Gautschi, M.; Rossi, D.; Benini, L. Customizing an Open Source Processor to Fit in an Ultra-low Power Cluster with a Shared L1 Memory. In Proceedings of the GLSVLSI '14, 24th Edition of the Great Lakes Symposium on VLSI, Houston, TX, USA, 21–23 May 2014; ACM: New York, NY, USA, 2014; pp. 87–88.

30. Gautschi, M.; Schaffner, M.; Gürkaynak, F.K.; Benini, L. 4.6 A 65 nm CMOS 6.4-to-29.2pJ/FLOP@0.8V shared logarithmic floating point unit for acceleration of nonlinear function kernels in a tightly coupled processor cluster. In Proceedings of the 2016 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 31 January–4 February 2016, pp. 82–83.
31. Loi, I.; Rossi, D.; Haugou, G.; Gautschi, M.; Benini, L. Exploring Multi-banked shared-L1 Program Cache on Ultra-low Power, Tightly Coupled Processor Clusters. In Proceedings of the CF '15, 12th ACM International Conference on Computing Frontiers, Ischia, Italy, 18–21 May 2015; ACM: New York, NY, USA, 2015; pp. 64:1–64:8.
32. Teman, A.; Rossi, D.; Meinerzhagen, P.; Benini, L.; Burg, A. Power, Area, and Performance Optimization of Standard Cell Memory Arrays Through Controlled Placement. *ACM Trans. Des. Autom. Electron. Syst.* **2016**, *21*, 59:1–59:25.
33. Rossi, D.; Loi, I.; Haugou, G.; Benini, L. Ultra-low-latency Lightweight DMA for Tightly Coupled Multi-core Clusters. In Proceedings of the CF '14, 11th ACM Conference on Computing Frontiers, Cagliari, Italy, 20–22 May 2014; ACM: New York, NY, USA, 2014; pp. 15:1–15:10.
34. Conti, F.; Palossi, D.; Marongiu, A.; Rossi, D.; Benini, L. Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms. In Proceedings of the 2016 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 14–18 March 2016; pp. 1201–1206.
35. Marongiu, A.; Benini, L. An OpenMP Compiler for Efficient Use of Distributed Scratchpad Memory in MPSoCs. *IEEE Trans. Comput.* **2012**, *61*, 222–236.
36. Alotaiby, T.N.; Alshebeili, S.A.; Alshawi, T.; Ahmad, I.; El-Samie, F.E.A. EEG seizure detection and prediction algorithms: A survey. *EURASIP J. Adv. Signal Proc.* **2014**, *2014*, 183.
37. Benatti, S.; Farella, E.; Benini, L. Towards EMG control interface for smart garments. In Proceedings of the 2014 ACM International Symposium on Wearable Computers: Adjunct Program, Seattle, WA, USA, 13–17 September 2014; ACM: New York, NY, USA, 2014; pp. 163–170.
38. Scholkopf, B.; Sung, K.K.; Burges, C.J.; Girosi, F.; Niyogi, P.; Poggio, T.; Vapnik, V. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Trans. Signal Proc.* **1997**, *45*, 2758–2765.
39. Bradley, A.P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognit.* **1997**, *30*, 1145–1159.
40. Wilkinson, J.H.; Bauer, F.L.; Reinsch, C. *Linear Algebra*; Springer: Heidelberger, Germany, 2013; Volume 2.
41. Delorme, A.; Makeig, S. EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *J. Neurosci. Methods* **2004**, *134*, 9–21.
42. Brunelli, C.; Campi, F.; Mucci, C.; Rossi, D.; Ahonen, T.; Kylliäinen, J.; Garzia, F.; Nurmi, J. Design space exploration of an open-source, IP-reusable, scalable floating-point engine for embedded applications. *J. Syst. Archit.* **2008**, *54*, 1143–1154.
43. Oberstar, E.L. Fixed-point representation & fractional math. *Oberstar Consulting Rev.* **2007**, *1*.
44. Hsu, C.W.; Chang, C.C.; Lin, C.J. A practical guide to support vector classification. *Tech. Rep. Taipei* **2003**.
45. Shoeb, A.H. Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2009.
46. Chandaka, S.; Chatterjee, A.; Munshi, S. Cross-correlation aided support vector machine classifier for classification of EEG signals. *Expert Syst. Appl.* **2009**, *36*, 1329–1336.
47. Acharya, U.R.; Molinari, F.; Sree, S.V.; Chattopadhyay, S.; Ng, K.H.; Suri, J.S. Automated diagnosis of epileptic EEG using entropies. *Biomed. Signal Proc. Control* **2012**, *7*, 401–408.
48. Nicolaou, N.; Georgiou, J. Detection of epileptic electroencephalogram based on permutation entropy and support vector machines. *Expert Syst. Appl.* **2012**, *39*, 202–209.
49. Rapoport, B.I.; Kedzierski, J.T.; Sarpeshkar, R. A glucose fuel cell for implantable brain–machine interfaces. *PLoS ONE* **2012**, *7*, e38436.