Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTRONICA, TELECOMUNICAZIONI E TECNOLOGIE DELL'INFORMAZIONE

Ciclo XXXII

Settore Concorsuale: 09/E3 - ELETTRONICA

Settore Scientifico Disciplinare: ING-INF/01 - ELETTRONICA

Optimized Biosignals Processing Algorithms for New Designs of Human Machine Interfaces on Parallel Ultra-Low Power Architectures

Presentata da: MONTAGNA FABIO

Coordinatore Dottorato **Supervisore**

Prof. COSTANZO ALESSANDRA **Prof. BENINI LUCA**

Esame finale anno 2020

ALMA MATER STUDIORUM - UNIVERSITY OF BOLOGNA

Optimized Biosignals Processing Algorithms for New Designs of Human Machine Interfaces on Parallel Ultra-Low Power Architectures

by

Fabio Montagna

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering
Department of Electrical, Electronic, and Information Engineering (DEI)

February 2020

“C’est le temps que tu a perdu pour ta rose qui fait ta rose si importante.”

Antoine de Saint-Exupry, Le petit prince

ALMA MATER STUDIORUM - UNIVERSITY OF BOLOGNA

Abstract

Faculty of Engineering
Department of Electrical, Electronic, and Information Engineering (DEI)

Doctor of Philosophy

by Fabio Montagna

The aim of this dissertation is to explore Human Machine Interfaces (HMIs) in a variety of biomedical scenarios. The research addresses typical challenges in wearable and implantable devices for diagnostic, monitoring, and prosthetic purposes, suggesting a methodology for tailoring such applications to cutting edge embedded architectures. The main challenge is the enhancement of high-level applications, also introducing Machine Learning (ML) algorithms, using parallel programming and specialized hardware to improve the performance. The majority of these algorithms are computationally intensive, posing significant challenges for the deployment on embedded devices, which have several limitations in term of memory size, maximum operative frequency, and battery duration. The proposed solutions take advantage of a Parallel Ultra-Low Power (PULP) architecture, enhancing the elaboration on specific target architectures, heavily optimizing the execution, exploiting software and hardware resources. The thesis starts by describing a methodology that can be considered a guideline to efficiently implement algorithms on embedded architectures. This is followed by several case studies in the biomedical field, starting with the analysis of a Hand Gesture Recognition, based on the Hyperdimensional Computing algorithm, which allows performing a fast on-chip re-training, and a comparison with the state-of-the-art Support Vector Machine (SVM); then a Brain Machine Interface (BCI) to detect the respond of the brain to a visual stimulus follows in the manuscript. Furthermore, a seizure detection application is also presented, exploring different solutions for the dimensionality reduction of the input signals. The last part is dedicated to an exploration of typical modules for the development of optimized ECG-based applications.

Acknowledgements

I am especially indebted to Prof. Luca Benini for his invaluable research support and academic advice during the entire period of my PhD. I specifically appreciated Prof. Davide Rossi and Dr. Simone Benatti's extensive personal and professional guidance and provision of precious knowledge. I special thanks to Prof. David Atienza Alonso and the ESL Lab from EPFL that hosted me for 6 months. I would like to thanks my lab-mates Victor, Alberto, Mattia, Emanuele, Manu, Tommaso, Beppe, Angelo, Marco, ..., and the entire PULP team, for all the coffee breaks and (of course) the great technical advises. In the end, I am grateful to my parents, siblings, relatives, and friends for all their psychological support without which I would not have been able to go through this challenging work.

Contents

List of Figures

List of Tables

Abbreviations

ADC	**A**nalog to **D**igital **C**onverter
AFE	**A**nalog **F**ront-**E**nd
AP	**A**ctive **P**otential
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
ASSR	**A**uditory **S**teady-**S**tate **R**esponse
BLE	**B**luetooth **L**ow **E**nergy
BT	**B**lue**T**ooth
CMOS	**C**omplementary **M**etal-**O**xide **S**emiconductor
CORDIC	**CO**ordinate **R**otation **D**ig**I**tal **C**omputer
CR	**C**ompression **R**atio
CS	**C**ompressing **S**ensing
DCT	**D**iscrete **C**osine **T**ransform
DMA	**D**irect **M**emory **A**ccess
DSP	**D**igital **S**ignal **P**rocessing
DT	**D**ecision **T**ree
DWT	**D**iscrete **W**avelet **T**ransform
ECG	**E**lectro**C**ardio**G**raphy
EEG	**E**lectro**E**ncephalo**G**raphy
EMG	**E**lectro **M**yo **G**raphy
EPSP	**E**xcitatory **P**ost**S**ynaptic **P**otential
ERP	**E**vent-**R**elated **P**otential
FD-SOI	**F**ully **D**epleted-**S**ilicon **O**n **I**nsulator
FE	**F**eature **E**xtraction
FFT	**F**ast **F**ourier **T**ransform
FIR	**F**inite **I**npulse **R**esponse

FLL	**F**requency-**L**ocked **L**oop
fMRI	**f**unctional **M**agnetic **R**esonance **I**maging
FPU	**F**loating **P**oint **U**nit
FTA	**F**requency **T**agging **A**nalysis
FWL	**F**ractional **W**ord **L**ength
GMM	**G**aussian **M**ixture **M**odel
GPIO	**G**eneral **P**urpose **I**nput **O**utput
HMI	**H**uman **M**achine **I**nterfaces
HPF	**H**igh **P**ass **F**ilters
IA	**I**nstrumentation **A**mplifier
ICA	**I**ndependent **C**omponent **A**nalysis
IoT	**I**nternet **o**f **T**hings
ISA	**I**nstruction **S**et **A**rchitecture
IWA	**I**nteger **W**ord **L**ength
JTAG	**J**oint **T**est **A**ction **G**roup
KNN	**K**- **N**earest **N**eighbor
LDO	**L**ow**D**rop**O**ut
LDA	**L**inear **D**iscriminant **A**nalysis
LUT	**L**ook **U**p **T**ables
LPF	**L**ow **P**ass **F**ilter
LSP	**L**omb-**S**cargle **P**eriodogram
MAC	**M**ultiply-**AC**cumulate
MCU	**M**icro**C**ontroller **U**nit
MEG	**M**agneto**E**ncephalo**G**raphy
NP	**N**ormalized **P**ower
PC	**P**rincipal **C**omponent
PCA	**P**rincipal **C**omponent **A**nalysis
PULP	**P**arallel **U**ltra-**L**ow **P**ower
QSPI	**Q**uad **S**erial **P**eripheral **I**nterface
RBPNN	**R**adial **B**asis **P**robabilistic **N**eural **N**etwork
ROC	**R**eceiver **O**perating **C**haracteristic
RTC	**R**eal-**T**ime **C**lock
SCM	**S**tandard **C**ell **M**emory

SIMD	**S**ingle **I**nstruction **M**ultiple **D**ata
SoC	**S**ystem **o**n **C**hip
SNR	**S**ignal-to-**N**oise **R**atio
SPI	**S**erial **P**eripheral **I**nterface
SRAM	**S**tatic **R**andom **A**ccess **M**emory
SSVEP	**S**teady **S**tate **V**isually **E**voked **P**otentials
SVD	**S**ingular **V**ector **D**ecomposition
SVM	**S**upport **V**ector **M**achine
TCDM	**T**ightlyCoupled **D**ata **M**emory
WPD	**W**avelet **P**acket **D**ecomposition

to my lovely family.

Chapter 1

Introduction

1.1 Thesis Context

Recent advances in bio-sensing and computing technology are enabling a wide range of cutting edge Human-Machine Interfaces (HMIs). Researchers have been working on HMIs to develop efficient devices for several purposes, among which the treatment of neurological pathology, monitoring and rehabilitation, to augment human abilities, restoring the daily life habits of a user. The fast improvement in the design of miniaturized and efficient electronic devices enables the development of unobtrusive implantable and wearable personal health care systems. In general, biosignals such as ECG, EEG, EMG (ExG) result from the physiological activity of the body, allowing to highlight biomedical parameters directly connected to the subject health status or simply to extract patterns that enable direct communication with the external environment.

ExG applications are widely used in various scenarios, ranging from consumer electronics (i.e. fitness, recreational) to medical-grade devices (i.e. patient monitoring, rehabilitation, prosthetic). For instance, ECG processing for the heart rate monitoring is employed in smartwatches for fitness, pacemakers [6] and Holter devices [7]. EEG based systems are becoming increasingly popular in medical applications for the treatment of neurological disorders, such as Parkinsons [8], epilepsy [9], spinal cord injuries [10], [11], [12] as well as attention loss, drowsiness [13] and autism detection [14]. Furthermore, implantable EMG recording has been extensively explored in research [15]; nevertheless, surface EMG devices for wearable systems are preferred thanks to their unobtrusiveness. It has been demonstrated that they are feasible for multiple applications, including the controllers of the upper limp prostheses[16] and hand gesture recognition systems for consumer Human-Machine interaction.

Due to the intrinsic variability of biosignals, in addition to an adequate signal conditioning and pre-processing phase, algorithms from the machine learning area are often necessary to reach high accuracy in recognition, as required by medical standards. However, this requires substantial computational capabilities and very low power consumption that only became suitable for wearable or implantable devices in the last years. For this reason, the first studies on ExG were performed offline on benchtop platforms [17], which offered all the required computational power. The advances in digital low-power design and efficient computational architectures [18], [19] made possible the design of real-time systems able to execute algorithms like advanced filtering [20], dimensionality reduction [19] and pattern recognition [21].

Notwithstanding, several challenges are still open for the design of efficient real-time systems for the processing of ExG signals. The solution to these challenges requires a multilevel approach, ranging from the design of Analog Front Ends (AFE) and digital processor to the system level architecture and algorithms. A highly efficient multi-core platform, designed for ultra-low power processing allows the execution of computationally demanding algorithms complying with real-time requirements, bounding the power consumption to a few mW using power management techniques coupled with voltage scaling up to near-threshold computing. Among the possible commercial and academic solutions, it is worth to explore the capabilities of the PULP platform, compared with the ARM Cortex M4, widely used in embedded devices. It has been largely demonstrated that the PULP-based architectures offer high computational power, but still remaining in the ultra-low power budget, making them feasible for a wide range of applications, going from IoT, drones control, image processing, and finally to biomedical applications.

1.2 Thesis Contribution

The main contribution of this dissertation is the definition of a methodology to develop algorithms for the processing of different kinds of biosignals, focusing the attention on cutting edge parallel ultra-low power architectures and the derived benefits compared to the use of commercial micro-controllers (MCUs). The thesis offers guidelines to efficiently target biomedical applications on embedded devices, typically characterized by having limited resources. In particular, the description focuses on how to take advantage of the architectural behaviors using both software and hardware optimizations, such as parallel programming, specialized hardware, and memory management. Parallel computing helps in solving tasks that require to respect strict real-time requirements, which are impossible to reach with the maximum operative frequency of a single processing unit. In the case on which a single processing unit is enough to respect the latency constraint, the use of multiple cores working at a lower (and more efficient) operative frequency can lead to a considerable energy saving. In some architectures, the exploration of specialized hardware oriented to solve simple operations in a few clock cycles, further improves the performance in latency and energy consumption. The memory management can be challenging in embedded devices, in particular for architectures that feature multi-level memories and for applications, which require frequent data transfers. For the architectures that feature a Direct Memory Access (DMA), the implementation of a policy to move data from one memory to another helps to reduce the overhead of the transfers and, thus, the latency.

Another contribution of this dissertation is the presentation of several case studies of different ExG applications, addressing challenges not only at system level but also at application level. Results in terms of execution time and energy consumption, with a detailed state-of-the-art (SoA) analysis, highlight the benefits derived from the use of software and hardware optimization applied in real applications. For instance, a challenge addressed in this thesis is related to the training of the algorithm for an EMG-based hand gesture recognition. This phase usually requires an off-line elaboration using an external device, not feasible for a fully embedded system, where several re-training sessions are necessary for the correct functionality of the device. In this scenario, the main contribution is the development of an algorithm suitable for real-time one-shot learning. To the best of my knowledge, this is the first time that a full EMG-based wearable system with online learning is being implemented. It has been demonstrated that the proposed solution is comparable with the well-known SoA method Support Vector Machine (SVM) and that the system can be scaled up efficiently to a higher number of channels for other more intensive ExG applications.

Focusing the attention on the EEG, during the acquisition the signal is prone to the noise coming from different sources such as power-line interference, movement and muscular artifacts. Considering as a possible task the detection of frequency as a response to an external stimulus, this noise significantly degrades the signal, which can reduce the system ability to detect very low frequencies.For this reason, for the noise that can not be removed using filters, the classical approach usually involves several sessions of visual inspection performed by a clinician to remove bad segments from the input data. The contribution in this field is the implementation of a fully automatic approach to remove bad segments from the input signals and, then, to perform all the processing for the detection of the target frequencies on-chip, providing immediate feedback to the user. Moreover, the EEG signals are usually acquired by multiple electrodes that cover the entire surface of the scalp of a subject. The number of electrodes ranges from 4 to 256, leading to a huge quantity of data. The embedded devices mainly feature small memories, with limited storage capacity. The processing of a huge quantity of inputs can lead to latency not compliant with the medical standards. Therefrom, dimensionality reduction algorithms help to remove all the redundancies in the data, drastically reducing the memory requirements and the latency of the processing. In this dissertation, two different approaches are compared, namely Principal Component Analysis and the Compressing Sensing (CS), showing a reduction of the computational complexity while maintaining the same accuracy.

1.3 Thesis Structure

The rest of the thesis is organized as follows.

Chapter 2 gives an overview of the commercial MCU (ARM Cortex M4) and the PULP platform, describing in detail the architectures targeted in this dissertation. Moreover, the same chapter shows a possible methodology that can be followed during the development of a variety of applications in several fields such as IoT, imagine processing, biomedical, etc. The suggested methodology can be considered as a guideline that helps the user during the developing process, giving some tips for the improvement of the efficiency. In particular, depending on the architecture, software and hardware optimization can be included in the implementation. In this chapter, a software optimization centered on the parallel paradigm is presented, with some guidelines on how to proceed during the parallelization of an algorithm. Then, hardware optimization is also presented, describing the RI5CY ISA extensions such as optimized instructions for loads and stores, ALU extensions, MAC operations, etc. The last optimization is related to memory management and how to efficiently handle the data transfers between the high latency external memory to the low latency internal one.

Furthermore, a collection of applications in the biomedical field are presented as case studies, showing the improvements derived by the integration of the optimization in performance. In particular, in Chapter 3, an EMG-based hand gesture recognition is implemented on embedded devices, based on two ML classification mechanisms. The first one based on the state of the art SVM and the second one on HD computing. Both approaches are compared in terms of execution time, energy consumption and training time.

In Chapter 4, several case studies related to the implementation of BMIs are presented. In this chapter, different kinds of pre-processing for the EEG signals are explored. Specifically, the first application is based on the FTA for the detection of the brain's response to an external visual stimulus. An automatic approach for the artifacts' removal is presented, that is compared to the classic approach that makes use of very complex algorithms (i.e. Independent Component Analysis), requiring manual inspection of the EEG traces performed by a clinician. Then, all the kernels that compose the final application are described, presenting results related to execution time and energy consumption of the final system. The second application targets the prediction of seizures in epileptic subjects and it is based on two different approaches are explored, differing from dimensionality reduction algorithm and feature extraction.

In Chapter 5, a description of the ECG signals is done, with an exploration of a collection of typical kernels that can be used for the development of a complete ECG-based system. We show results derived by the parallel computing on these kernels in terms of execution time and energy consumption.

Finally, in Chapter 6 conclusion and feature works are drawn to understand the direction of the research for future improvements.

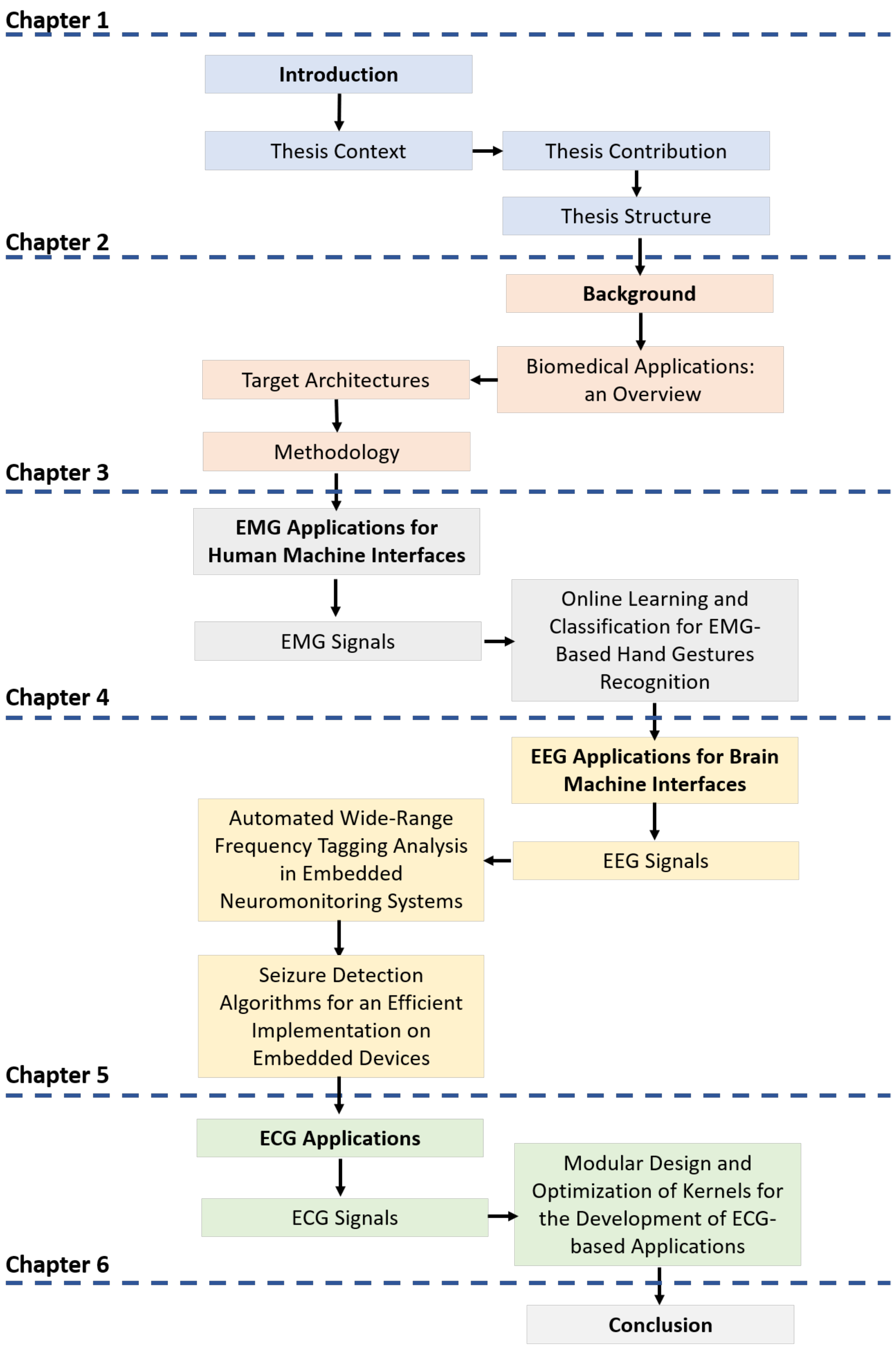

Chapter 1
Introduction
Thesis Context
Thesis Contribution
Thesis Structure
Chapter 2
Background
Biomedical Applications: an Overview
Target Architectures
Methodology
Chapter 3
EMG Applications for Human Machine Interfaces
EMG Signals
Online Learning and Classification for EMG-Based Hand Gestures Recognition
Chapter 4
EEG Applications for Brain Machine Interfaces
EEG Signals
Automated Wide-Range Frequency Tagging Analysis in Embedded Neuromonitoring Systems
Seizure Detection Algorithms for an Efficient Implementation on Embedded Devices
Chapter 5
ECG Applications
ECG Signals
Modular Design and Optimization of Kernels for the Development of ECG-based Applications
Chapter 6
Conclusion

Figure 1.1: Thesis structure diagram.

Chapter 2

Background

2.1 Biomedical Applications: an Overview

Traditionally, the monitoring of the signals acquired from the human body has been done in the laboratories under the supervision of clinicians. Researchers have been trying to improve people's quality of life, developing devices able to produce reliable data that can be used independently outside the medical environment and analyzed by clinicians to track health status or to detect specific events, providing immediate feedback. Moreover, the elaboration of all the information collected during the monitoring is also essential for off-line explorations to improve the knowledge of human body's behaviors. In the last years, with the fast progress in technologies, huge improvements have been done in biomedical devices. Nonetheless, there are still a variety of possible scenarios that require further improvements. For this dissertation, we explore possible applications for an efficient EMG-based hand gesture recognition, a frequency tagging analysis using EEG signals and a seizure detection algorithm. Before going into deeper details in the next chapters, an overview of each application scenario can be useful to have a general idea about the topics and the possible issues and challenges we might deal with.

Since interacting with hands represents one of the most intuitive ways to enable human-to-human or human-to-machine interactions, hand gesture recognition is the key element in designing solutions that can enable natural and advanced ways of communication between objects and users in many domains, in the wake of the IoT growing trend. The two major approaches used for hand gesture recognition are based on the processing of information coming from video cameras [22] and from muscular activity [23]. Video-based hand gesture recognition relies on computer vision techniques, which recognize users' hand gestures in a scene and decode the hand gestures using pattern recognition algorithms. Although this approach can decode a wide number of gestures, it suffers from ambient illumination variability and possible obstacles in the line of sight. Furthermore, it requires a bulky and fixed setup.

Another viable approach is inspired by hand prosthetic systems, where the muscular activity detected by the electromyographic (EMG) signals is used to decode the user's intention. Commercial prosthetic systems [24, 25] decode predefined bursts of muscular contractions into a set of gestures (i.e. 2 consecutive contractions mean hand closures, 3 consecutive contractions stands for hand opening, etc.). Such a method is highly reliable and amenable to implement on a wearable system. However, it requires a long learning curve and high levels of concentration of the user since it is a non-intuitive interface [26]. An approach that is gaining traction is based on machine learning (ML) techniques to analyze EMG signal patterns during muscular contractions. It has been demonstrated that algorithms such as Linear Discriminant Analysis (LDA) [27], Artificial Neural Networks (ANN) [28], Support Vector Machines (SVM) [29], Recursive Least Square [30], Hidden Markov Models (HMM) [4], Naive Bayes [31], Independent Component Analysis (ICA) [32], as well as Convolutional Neural Networks (CNN) [33] reach high level of accuracy, restoring a natural gesture recognition both in prosthetic and consumer Human Machine Interaction (HMI) scenarios.

Using such machine learning algorithms allows tailoring the recognition strategies to the physiological characteristic of the users. However, EMG-based gesture recognition is strongly dependent on the subject (i.e. it depends on subject-specific characteristics such as muscular mass, skin thickness, strength of the mean voluntary contractions), and classification algorithms need a training set for each user. Furthermore, EMG setup is intrinsically variable [34–36] because of fiber crosstalk, skin perspiration, small movements of the skin-to-electrode interface, power line interference and donning/doffing. As such, small changes of the EMG traces can hinder pattern recognition, degrading the system performance down to unacceptable levels [37]. An orthogonal approach consists of extending the training dataset, integrating it with samples coming from multiple sessions to gain up to 20% recognition accuracy [36]. A major drawback of increasing the size of the training dataset or repeating the algorithm training lies into its high computational requirements. For instance, SVM training minimizes a given cost function by solving a convex optimization problem, while in ANN [38] back-propagation requires many iterations to converge. Furthermore, training sessions require a PC or a graphical interface as well as human intervention to perform thresholding and labeling operations [39]. Developing a system capable of "one-shot" learning based on a computationally efficient and non-iterative algorithm for online training has the potential of significantly improving HMIs based on EMG signals.

Another important branch of research involves EEG signal processing for neurological applications. The advances in cognitive neuroscience have been giving us the possibility to interface directly with the human brain. Moreover, neurological disorders affect nearly one billion people, a considerable percentage of the world population [40]. The estimated economic costs were more than 2 trillions of USD in 2010 [41], with a high social impact. Despite the fact that effective treatments are available, a significant part of the population is untreated, because of inadequate healthcare infrastructure, lack of trained staff, effective diagnostics and screening tools. Among diagnostic and screening techniques, Electroencephalography (EEG) analysis and instrumentation is an established standard, since it directly records the electrical field generated by neural activity with a set of electrodes distributed on the head surface (scalp) [42]. Thanks to its effectiveness, non-invasiveness, low cost and portability, EEG is one of the most used techniques for investigating brain function and pathology, both in clinical settings and scientific research [43] [44] [45].

One effective use of the EEG signals is to analyze brain responses to specific stimuli. The most popular method to measure the EEG response to a stimulus is to average the EEG signal across several stimulus presentations (Event-Related Potential, ERP [46]). Since the neural activity unrelated to the stimulus typically fluctuates within the same time scales of the stimulus related activity, a high number of stimulus presentations is needed to average it out and extract the stimulus-related response. An alternative technique, frequency-tagging (FT), has been developed to reliably measure stimulus-related EEG responses in a much shorter time. This technique exploits the property of the brain activity to respond to a visual or auditory stimulus presented periodically at a specific (i.e. ”tag”) temporal frequency by resonating at the same frequency during the stimulation period [47, 48]. This effect is manifested in the EEG recordings by a sharp peak in the power spectrum of the signal at that specific tag frequency. Since the EEG ongoing activity is broad-band in frequency, the stimulus-related response in the frequency domain is very easily discriminated from the stimulus-unrelated activity, yielding a much higher SNR than the one obtained with ERPs. Moreover, since most EEG artifacts (eye movements, blinks) are also broad-band in frequency, FT is more robust than ERP to artifacts and requires a lighter artifact rejection procedure.

Thanks to the short time needed to have a reliable response, FT has always been used in clinical settings to test the integrity of sensory areas [42] in the visual domain (classically defined Steady-State Visually Evoked Potentials - SSVEP [48]) and in the auditory domain (classically defined auditory steady-state responses - ASSR [47]), by presenting simple visual and auditory stimuli at relatively high frequencies (10 to 40 Hz), since sensory systems are most responsive at those frequencies. Since the amplitude of both ongoing EEG activity and eye-movement-related artifacts are relatively low in this frequency range, fast, automatic techniques have been developed to rapidly extract the frequency responses, in particular in the field of SSVEP-based BCI systems [49].

Most recently, the use of FT has been extended to investigate the neural responses related to higher-level perceptual or cognitive functions, such as attention [50], speech [51] or face [52] recognition. Since such functions require long neural processing, stimulation frequencies in a low-frequency range (0.5-6Hz, hereafter referred to as ”low-frequency”) have to be used. It has been shown that FT is still effective at those frequencies [51]. Because of their performance in obtaining a stimulus-specific neural response in a short time, very recent FT designs based on stimulations in the low-frequency range have been successfully used as a tool for investigating the neural basis of cognitive development in very young children [53], [54]. Given these positive results, FT is a promising tool for testing brain function in clinical settings and/or with vulnerable populations as newborns or aged people.

However, for FT designs based on the low-frequency range, EEG ongoing fluctuations and artifacts are much more relevant than in the typical frequency range of SSVEP (>6Hz). Therefore a significant human intervention is needed to clean the data from artifacts and extract the response. Typically, the EEG traces are acquired and processed off-line on bench-top platforms, since most of the techniques to analyze brainwaves requires heavy computational processing and visual inspection from technicians or medical staff. This procedure implies non-negligible cost and time for an accurate analysis, due to the need for data transfer, off-line visualization, manual inspection and tagging. For instance, artifacts identification and removal require a combination of the visual evaluation of the EEG trace and of algorithmic techniques such as Independent Component Analysis (ICA), digital filtering, interpolations and averaging. Automating these analyses for FT stimulation in the low-frequency range would dramatically improve research and diagnosis, enabling the design of extensive screening systems for many neural disorders. Luckily, we are witnessing the massive technological trend of embedded wearable applications, that are quickly becoming pervasive [55], led by the constant growth of the health-care market and by the boost of digital technologies integration [3]. This trend paves the way for designing embedded, energy-efficient systems for biosignal processing based on advanced algorithmic techniques [20, 56, 57].

Another effective application using the EEG signals in BMI is represented by the detection of seizures in epilepsy. Among the neural diseases, 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 [58] 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 in treating epilepsy while it does not present collateral effects [59]. 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 [12]. 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 multi-channel systems [60–62] 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, 63]. 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-function ASIC designs, implementing feature extraction and pattern recognition algorithms for classification [64, 65]. 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.

From the algorithmic point of view, this issue has been addressed by applying dimensionality reduction algorithms, such as PCA and CS at the beginning of the processing chain. Although PCA approach has been exploited in several works elaborating the data offline [61], 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. To drastically reduce the complexity of the preprocessing, a promising approach is to leverage the Compressed Sensing (CS) techniques. For instance, in [66], the authors implemented a seizure detection algorithm on a low-power domain-specific many-core platform. As opposed to multi-channel approaches based on traditional dimensionality reduction algorithms[5], in [66] compressed sensing reduced the computational complexity but with poor classification accuracy. On the other hand, it can be demonstrated that coupling preprocessing based on compressed sensing with powerful feature extraction and classification techniques allows achieving classification performance results similar to traditional approaches like PCA coupled with lightweight execution typical of CS approaches.

2.2 Target Architectures

Currently, the majority of the embedded devices able to operate in ultra-low power modes, last several years using a small battery, elaborating specific low complexity tasks and leaving the heavy computation to external devices. The aim of these devices is to consume the least possible to the detriment of high performance. In fact, they are capable to operate at very low range frequencies (from a few tens of kHz, up to a few MHz), sometimes at sub-threshold voltages. Nowadays, a new trend is changing the way in which the raw signals acquired by the sensors are elaborated. Instead of transmitting the data to an external device or the cloud, the data are elaborated close to the sensor, leaving the data transfer only for highly compressed and informative data such as classes, trigger events or alarms. This approach paves the way to new challenges based on the development of heavily optimize algorithms, remaining in the mW-range power envelope for always-on battery-powered embedded devices.

Several solutions have been proposed in the last years to increase the performance requirements for these devices. One solution has the aim of increasing the operative range of low power processors to satisfy higher computational complexity, maintaining a reasonable efficiency. The problem occurs when the constraints of the application (i.e. latency) are too tight and the system is forced to operate out of the sub/near-threshold operative voltages, leading to a drop in energy efficiency. Another solution implies the use of specialized hardware to accelerate the computation, leaving the general-purpose processor for other tasks. Recently, a different solution is becoming popular and derives from parallel near-threshold computing. With this solution, it is possible to exploit the benefit of the near-threshold computing, maintaining high-performance thanks to the multi-core execution and the flexibility leveraging software programmable processors.

Typically, the hardware requirements for the processing of biosignals in biomedical applications depend on the complexity of the algorithms, the scope of the embedded device (i.e. implantable or wearable) and the application itself. First, to start the processing we need to collect inputs from the human body. To ensure this, the architecture we are targeting should be able to support an Analog Front End for the acquisition of multiple channels. As an example, a good choice is the ADS1298 from TI, the de-facto standard used in biopotential acquisition platforms. It presents a very favorable trade-off between performance and power consumption since its 3 V single supply does not require step-up DC/DC conversion of the battery voltage without significantly affecting noise performance.

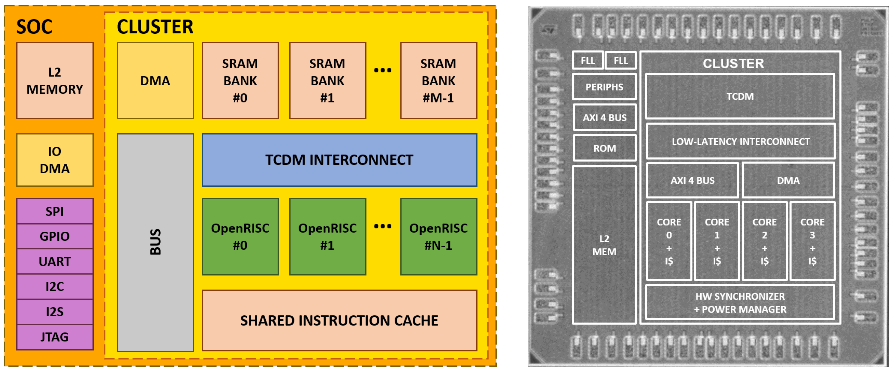
The size of memories is another important aspect we should take into account. For some ExG applications, we need to store a huge quantity of data before starting the final elaboration (i.e. coefficients, weights, etc.) or simply the input vector. Furthermore, we may need to store the outputs for the next iteration (if required) at the end of the elaboration. Usually, embedded devices feature a small size memory directly connected to the processor (to the cluster in the case of multi-core architectures), and another external memory. A useful component that should be featured by the architecture is the DMA to handle the data transfers between memories in a very efficient way.

After ensuring an efficient processing unit feed, we should think about how to represent this information and elaborate them accordingly. It can happen that the application does not, or just in part, require floating-point arithmetic operations that are double-precision operations and, thus, more complex in terms of computation. In this case, we can decide if it is convenient to feature an FPU, leading to larger energy consumption, or to simulate these floating-point operations in software, but increasing the computation. Sometimes, the best solution is to convert the floating-point implementation into fixed-point. This is not always possible and it has been demonstrated that, in a complete application that includes several complex kernels, a hybrid approach (to switch from floating-point to fixed-point depending on the kernel) can be more effective[19]. Some architectures feature dedicated hardware to compute basic operations that can be used to dramatically optimize the execution of the code, such as MAC instructions, pop-counters, SIMD instructions, etc. For all these considerations, we should carefully choose the target architecture, based on the specific requirements of the application we are working on.

2.2.1 ARM Cortex M4

The ARM Cortex M4 processor, designed for efficient DSP and energy efficiency, is the most commonly used commercial MCU for embedded systems. The processor is based on Harvard architecture with three-pipeline stage and branch speculation, implementing 32-bit RISC ISA. It also includes the entire 16-bit $Thumb^{®}-1$ and 16/32-bit $Thumb^{®}-2$ instruction sets. It features an FPU that can be activated for fast floating-point operations. The ARM Cortex M4 MCU works at an operating frequency of 168 MHz and 1.85 V. Implemented in a 90nm technology with low power processing of 33 uW/MHz, it addresses the challenge of maintaining low dynamic power constraints. This low power consumption leads to a longer battery lifetime, which is critical in embedded devices for a wide range of applications.

When operating with critical tasks and interrupt routines, a known number of cycles manages these requests, enhancing determinism. It features non-maskable interrupts(NMI) and up to 240 physical interrupts, with different priority levels (from 8 to 256). It supports sleep modes to reduce energy waste, with up to 240 wake-up interrupts and optional retention mode, integrating Wait For Interrupt (WFI) and Wait For Event (WFE) instructions and Sleep On Exit capability. It includes extended single-cycle 16/32-bit multiply-accumulate (MAC), dual 16-bit MAC instructions, optimized 8/16-bit SIMD arithmetic and saturating arithmetic instructions. For debug, it features a JTAG and Serial-Wire Debug (SWD) ports, with up to eight breakpoints and four watchpoints. The processor also has a Memory Protection Unit (MPU), up to 8 sub-regions. This design is used to protect the memory content, preventing access to privileged application data. Different buses interconnect the components of the SoC, providing data transfer management. It features a DMA for efficient management of the data transfers (SRAM, FLASH memory), avoiding occupancy of the processor.



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

Figure 2.1: 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**).

2.2.2 PULP platform

PULP platform is an open-source project born by the collaboration between the University of Bologna and the Eth Zurich. It is a many-core platform (Figure 2.2a) that can be based on OpenRISC or RISC-V Instruction Set Architecture (ISA). It is able to operate in a large range of operative voltages and frequencies, obtaining a high level of energy efficiency for different kinds of applications even with huge computational complexity (IoT, biomedical, image processing, etc). Several versions of the PULP architecture are available and they differ from the technology (CMOS, FD-SOI), form factor (65, 40, 28nm), number of processing units, and memory size.

The PULP fabric is integrated into a SoC that can feature an off-cluster (L2) memory; peripheral accesses are managed by a low power DMA (Direct Memory Access) 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 [67]. 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 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 [68]. 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).

In the exploration of the applications presented in this dissertation, different architectures related to the PULP platform were considered. The selection of the architecture was dictated by the requirements of the applications such as memory usage, the computational complexity of the algorithm, information representation (i.e. floating-point, fixed-point).

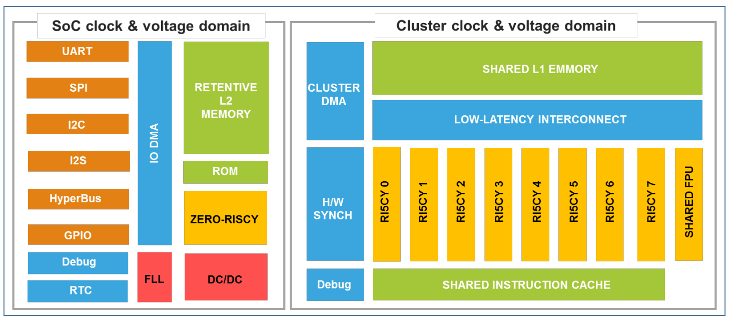
2.2.2.1 PulpV3 Architecture

The first architecture considered in this thesis is PulpV3, a 28-nm UTBB FD-SOI technology [69]. The computational engine of the SoC is a cluster with a parametric number (2–16) of cores (Figure 2.2a). The processors are based on a power-optimized four-pipeline stage micro-architecture implementing the OpenRISC ISA [70], 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 [71], 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 much higher energy efficiency [72].

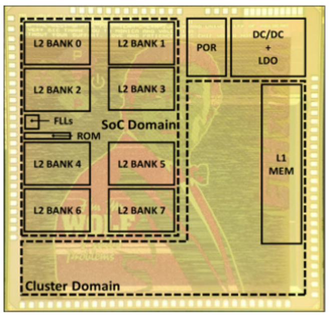
The cluster relies on an L1 explicitly-managed multi-banked Tightly-Coupled Data Memory (TCDM) for data access, avoiding memory coherency overhead of data caches and greatly increasing area and energy efficiency. Each logical bank can be implemented as a heterogeneous memory, either composed of SRAMs 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 [73]. 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.

2.2.2.2 Mr.Wolf Architecture

Mr.Wolf [74] is a CMOS 40nm technology. The SoC, shown in Figure 2.2, is a multi-core programmable processor coupling an advanced MCU controlled based on a tiny (12 Kgates)



(**a**)



(**b**)

Figure 2.2: A general view of the Parallel Ultra-Low Power (PULP) architecture (**a**) and the layout of the Mr. Wolf chip used for performance and power characterization (**b**).

RISC-V processor (zero-risky) [75] accelerated by a powerful 8-processors compute cluster leveraging the flexible and powerful DSP extensions available on the RI5CY processor [75]. The SoC contains a full set of peripherals, including a Quad SPI (QSPI) interface suitable to connect to a multi-channel EEG acquisition system. Data transfers from peripherals are autonomously managed by a multi-channel I/O DMA to minimize the amount of interactions with the controlling core when performing IO. 512 kB of L2 memory are available on the SoC, accessible from the parallel cluster through a dedicated DMA controller.

The cluster is equipped with a single cycle latency, multi-banked L1 memory, enabling shared-memory parallel programming models such as OpenMP. Two floating-point units (FPU) are shared among the 8 processors of the cluster, necessary to accelerate computations when highly dynamic data is present such as in EEG processing algorithms. Fast event management, parallel thread dispatching, and synchronization are supported by a dedicated hardware block (HW Sync), enabling very fine-grained parallelism and hence high energy efficiency in parallel workloads. To maximize power efficiency, the SoC contains an internal DC/DC converter that can be directly connected to an external battery that can deliver voltages in the range of 0.8 V to 1.1 V. When the system is in sleep mode this regulator is turned off and a lowdropout (LDO) regulator powers the real-time clock fed by a 32 kHz crystal oscillator, which controls programmed wake-up and, optionally, part of the L2 memory allowing retention of application state for fast wake-up. When in deep sleep the current consumption is reduced to 72 μW (from VBAT) assuming the RTC is active and no data retention, and 108 μW assuming full L2 retention.

2.3 Methodology

The purpose of the explorations described in this dissertation is to demonstrate the advantages of implementing algorithms for the processing of biosignals on cutting edge parallel ultra-low power architectures, exploiting parallel computing and heavy software and hardware optimization. The aim is to maximize energy efficiency (i.e. battery lifetime), maintaining high accuracy and low latency in the response. The architectures adopted for embedded implementations are characterized by having limited resources to improve the form factor and energy consumption. For this reason, it is necessary to adopt a methodology to take advantage of these architectures optimizing resource usage. To do that, there are several aspects we have to take into account, such as how to obtain a high degree of parallelism in many-cores implementations, optimizing the execution using software or hardware optimizations, and correctly managing the memories available on the SoC.

To efficiently implement an application on an embedded device, the first step is a careful and detailed feasibility study based on the application itself and the possible target architectures. This can be done dividing the application into small pieces (kernels), analyzing all the kernels separately and all the resources available on the architectures, trying to understand which part can be optimized and the possible issues that may occur. The first issue can be represented by the memory requirements of all the kernels that compose the application. It is useful to understand exactly all the vectors and matrices we need to allocate in memory (both statically and dynamically). Moreover, another important aspect to consider is the best representation of the data. This analysis allow to understand whether the integer implementation (or eventually the fixed-point representation) is feasible for our elaboration or if we need to use floating-point (FP) operations. In the last case, if the architecture does not feature a Floating Point Unit (FPU), we should consider the impact on the performance of the FP software simulations, not negligible in intensive FP computations' kernels. With these considerations, we can already make a choice about the architecture on which we want to target our processing chain.

Then, a good practice is to first implement all the processing chain on a scripting tool (i.e. MATLAB) as a golden model to carefully check the theoretical correctness of our implementation (numeric results, accuracy, etc.) and also to check the results obtained with the embedded implementation. The next step is to start the real implementation and to check the results of each kernel that compose the final application in C code, taking in mind all the possible optimizations we can do and predisposing the code we are writing accordingly. In this way, it would be easier to port the code in the architecture we are targeting. When we have a stable version, we can start the porting on the target architecture, addressing all the possible optimization to speed-up the execution.

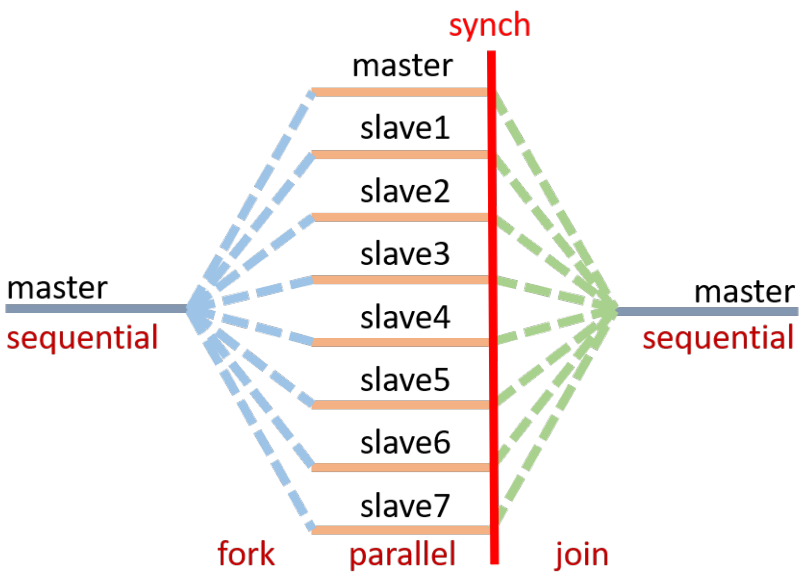


Figure 2.3: Fork/Join paradigm.

2.3.1 Software Optimization: Parallelization Paradigms

2.3.1.1 Programming Model and Toolchain

OpenMP [76] provides a programming model for shared memory multiprocessing architectures. Based on directives (C/C++ and Fortran), which are resolved at compile-time into low-level calls for the specific run-time. It relies on a fork/join parallel execution model (Fig. 2.3) guided by directives. The execution of the program starts with a single thread (the *master*). When a parallel construct is encountered, $n-1$ additional threads (*slaves*) are recruited into a parallel team. Work-sharing constructs are employed to specify how the parallel workload is distributed among the threads. When the parallel region ends, all the threads reach a barrier (red line in Fig. 2.3) for synchronization. Then, the master continues the execution in sequential until, eventually, it reaches a new parallel region.

The PULP platform relies on OpenMP 3.0 parallel library that operates on top of the GCC 7.1 programming toolchain. The OpenMP implementation is based on a highly-optimized bare-metal library [77], 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.

Before distributing the workload among the cores for parallel computing, it is important to evaluate different aspects of the implementation such as the data dependency, the computational complexity of the parallelizable sections, and the expected speed-up. There are two main type of parallelism, data parallelism and task parallelism. The data parallelism implies to split the data among the cores that execute more or less the same instructions, but on a different portion of the data. The first important evaluation is to check if there is a dependency between each iteration. It gives an idea about the degree of parallelism that is possible to obtain from the kernel we are considering. In particular, if the i-th iteration needs the data produced in the (i-1)-th iterations, the results obtained by the parallel execution of this *for* loop will be incorrect. One solution could be re-write the *for* loop in such a way that there is no dependency between iterations. If it is not possible, the *for* loop has to be executed sequentially. In the task parallelism, data are not divided among the cores, but different tasks are applied to the same data at the same time. Ideally, if our function is 100% parallelizable, the expected speed-up should be equal to the number of cores we are using for the elaboration. In reality, the speed-up is always lower than the ideal because we add some overhead to divide the workload and start the execution in parallel. The overhead is due to the code that has to be executed sequentially. Estimating the percentage of the sequential and parallel section, we can compute the maximum reachable speed-up using the formula:

$$S = \frac{1}{(1-p) + \frac{p}{N}} \tag{2.1}$$

where p is the percentage of the parallelizable sections over the total, and N is the number of cores we are using.

Another problem is represented by the computational complexity of the parallel section. For instance, if the workload of a *for* loop is small, there can be a significant overhead that can be detrimental for the speed-up of the system.Moreover, another aspect that should be taken into account is the workload in each iteration, and the number of iterations needed to complete the execution. It can happen that some cores may complete the work before the others and, then, they have to wait until all the cores reach the synchronization barrier to continue the execution - this is called workload unbalanced. If the workload unbalanced is due to a variable number of instructions that have to be executed in each iteration, it is sufficient to change the way to distribute the workload among the cores, passing from a static to a dynamic scheduling - the work is distributed dynamically to the cores that are in idle state. In other cases, this is not possible because the workload unbalanced is due to a number of iterations which is not a multiple of the number of cores. In this case, all the cores should wait for the other cores that are still working, even if their work is already finished.

Based on how often the cores need to synchronize or communicate with each other, the parallelism can be defined as fine-grained parallelism when several barriers are essential for the synchronizations, coarse-grained parallelism if there is no need of many synchronization points, and embarrassing parallelism if the barriers are sporadic or totally absent.

2.3.2 Hardware Optimization: RI5CY ISA Extension

Sometimes, some particular features of the hardware we are targeting can help in the optimization process. For instance, the processors inside the cluster of some PULP architectures feature the RI5CY ISA. Starting from the standard RISC-V ISA, the RI5CY supports multiple additional instructions used to further optimize performance. These extensions include post-incremental load and store instructions, hardware loops, ALU extensions such as bit manipulation instructions, and vectorial instructions, and Multiply-Accumulate (MAC) operations. The post-incremental load and store instructions perform a load or a store operation while increment the address used to access the memory. This approach reduces the number of instructions that are normally required when executing the code with the standard data access pattern. This is done attaching the address increment instruction to the memory access instruction. Moreover, the overhead due to the execution of a *for* loop is reduced combining this extension with the well-known hardware loops - the instruction inside the loop are stored in registers, avoiding to fetch the same instructions again.

The bit manipulation operations are very beneficial to optimize a code that requires several component-wise operations. For instance, if our application is based on the manipulation of binary vectors, the integration of these operations can drastically speed-up the computation. There are several operations we can use for this purpose such as the instruction to insert/extract a bit in/from a variable, to count the population in a variable (bits set to 1), the position of the first or the last bit set to 1 in a variable, etc. The vectorial operations perform instructions in a SIMD-like manner on multiple sub-word elements at the same time - these instructions are available for 8 or 16-bit operations. Among the available operations, there are several implementations to perform the dot product, to find the minimum or the maximum element in a vector, sum, subtract, and so on. The supported MAC operation is a 32-bit x 32-bit multiplier with a 32-bit result, performed in a single cycle. It also supports an extension for half-word multiplications with a post-multiplication shift (optional).

2.3.3 Memory Management Optimization: Double Buffering

As already mentioned in Section 2.2.2, PULP features two main memories, one inside the cluster (L1), which is directly interconnected with the processors, and another one outside the cluster (L2). The size of the L1 memory is smaller compared to the size of the L2 - for instance, Mr. Wolf features an L1 of 64kB and L2 of 512kB. Moreover, the L1 is a non-retentive memory,

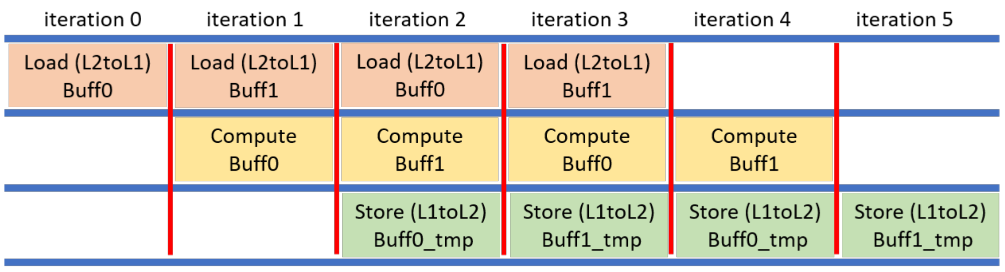


Figure 2.4: Double Buffering.

thus not feasible to store data that should be maintained for the entire lifetime of the application or if we need to store temporary data that do not fit in the L1 memory. Furthermore, for some applications and some architectures, using power management techniques improves the performance in terms of energy consumption, extending the battery life extensively. In some architectures, it is possible to scale down the voltage supply or the frequency, saving energy. Another approach is to put the SoC (if supported) in sleep mode avoiding unnecessary energy consumption, using the uDMA for the acquisition of the data. The uDMA is directly interconnected to the L2 memory and, when all the data required by the elaboration are available, the SoC wakes up and starts the elaboration on the entire acquired window. From the cluster point of view, the L1 memory has a low latency - it can be accessed for a load/store operation in one clock cycle, while the L2 memory requires a higher latency to be accessed - around 15 clock cycles. To avoid the overhead derived from the latency to access the L2 directly with the cluster, it is possible to use the DMA to transfer the data from the L2 memory to the L1 memory (or vice versa), implementing a double buffering policy. In this way, it is possible to mask the latency derived from the transfers between memories with the computation activity of the cluster.

Fig. 2.4 shows an example of a possible implementation of the double buffering policy. To correctly implement this technique, we first need to allocate two different buffers in the L1 memory. Then, we can easily switch between the two buffers using one of them to store the data coming from the DMA and the other one to elaborate the data transferred by the DMA in the previous iteration. The double buffering is composed of a given number of iterations driven by the switching policy between the buffers. The number of iterations depends on the computational complexity of the kernel we are considering, the data type and the transfer speed. To make all the process efficient, we should carefully tune the size of the buffers depending on the time required to complete the transfer and the time to elaborate the data inside the buffer, in such a way that when the cluster completes the elaboration of the chunk contained in the buffer, the other buffer is already full and ready to use.

The latency derived from the transfer of the first chunk of data is inevitable. As soon as the first transfer is complete, we switch the buffers and do a new call to the DMA for the transfer of the next chunk of data in the second buffer, while the cluster elaborates the data inside the first one. Sometimes, there is the need for transferring back the data inside the buffer. For instance, when the data inside the buffer change and we need to maintain the changes in memory for a successive elaboration, or simply when we need to keep the outputs for future processing. In this case, we should consider the time required to write back the data after each elaboration.

Chapter 3

EMG Applications for Human Machine Interfaces

3.1 EMG Signals

3.1.1 Signal Description and Acquisition

The EMG signal measures the electrical potential conducted by the muscular tissues, which represent the electrical activation of the muscle fibers. The electrical potentials are called muscle action potentials and they derive by the passage of Na+ ions and K+ ions through the nerve cell membranes, which generate the message sent by the brain for the muscle contraction. The Na+ ions flow pass along the cellular membrane through the Na channels, generating a depolarization and thus, to an electrical impulse that propagates through the target muscle cells. The flow of the Na+ ions into the nerve cell causes a release of Ca++ ions, leading to a cross-bridge binding and the muscle sarcomere contraction. The action potentials can be acquired using electrodes directly placed on the skin surface of the subject. The acquired EMG signals are the result of all these potentials of the cells. In this procedure, there are two issues represented by the signal-to-noise ratio - the ratio of the energy contained in the acquired EMG signal and the energy in the noise signal - while the second one is the distortion of the signal, when the frequency component of the EMG signals is somehow altered.

Typically, EMG signals are acquired through active sensors, composed of two conductive plates connected to the input of a differential amplifier. Furthermore, a conductive gel is applied to avoid a high level of impedance generated by the contact between the electrode and the skin. The amplitude of the EMG signals varies in the range [-10,+10] mV and its maximum bandwidth is around 2 kHz. These values depend on the distances between the sensor and the exact site of the contraction. Several disturbances affect the EMG signals, such as the power line noise, skin perspiration, etc.

Among the commercial solutions, Ottobock sensors are widely used in research and industrial applications. These sensors have a full-analog signal conditioning by a discrete bandpass filter, an instrumentation amplifier (IA) with a high gain stage and an offset cancellation feedback circuit that requires the use of a dedicated metal plate as the reference electrode for each sensor. The main disadvantages of using these sensors are the signal bandwidth reduction, limiting the possibility to extract features in the frequency domain and the high costs. An alternative is represented by passive electrodes, which allow avoiding the analog conditioning circuitry, moving this part to the on-board digital processing. In this way, it is possible to use low-cost electrodes, but still maintaining the same signal quality as in the active sensors.

3.2 Online Learning and Classification for EMG-Based Hand Gestures Recognition

In this chapter, several key points will be addressed and described in details, such as:

- The development of an algorithm suitable for real-time "one-shot" learning. To the best of our knowledge this is the first time that a full EMG-based wearable system with online learning is being proposed.
- Implementation and optimization of the algorithm on a parallel-ultra-low power architecture (i.e. Mr. Wolf).
- Design of an integrated wearable interface for gesture recognition embedding the Mr. Wolf SoC, extensively tested on a dataset acquired online on 10 subjects, and 11 gestures.
- The evaluation of the proposed system with respect to other methods for embedded gesture recognition, including a comparison with a state-of-the-art (SoA) method such as SVM [29].
- The exploration of the system scalability with respect to the number of electrodes available, following the general trend of bio-potential pattern recognition systems [78].

3.2.1 Related Work

In recent years, several systems have become available to acquire and process EMG signals for hand gesture recognition. This task requires a multi-modal approach which involves sensor interfaces, computational platforms and algorithms. The sensor interfaces are mostly based on commercial SoA Analog Front Ends, specifically designed for bio-potential acquisition. Regarding computational platforms and algorithms, top performance classification methods rely on supervised machine learning algorithms to reliably classify acquired data, such as SVM, LDA [79] or ANN [28]. The recognition accuracy of these algorithms is above 85%, and they are implementable on wearable platforms [39]. Despite some other methods, like RLS [30], can solve multiclass problems with negligible computational overhead, deterministic training time, and performance comparable to the aforementioned algorithms, in this work we performed a quantitative comparison with SVM, which represents the SoA algorithm and widely accepted baseline framework for EMG-based pattern recognition, already tested in several embedded implementations [79–84].

However, the major drawback common to all these approaches is related to the robustness of the setup, since the EMG signal is intrinsically affected by high variability due to both physiological and environmental factors. Once the training phase is performed on a given dataset, the algorithm does not generalize if signals somehow drift. Therefore, the reliability of such systems would greatly benefit of a fast real-time training, which can be repeated when needed. Unfortunately, in these approaches, the training phase is based on minimization of convex costs functions [85], backpropagation [86] or eigen decomposition [87]. Some of these techniques are iterative, with a convergence time that depends on the number of iterations required to minimize the error cost function. Furthermore, their computational and memory requirements severely hamper their implementation on resource-constrained platforms. For instance, LDA, which has a training faster than SVM and ANN [36], has time complexity of $O(mnt + t^3)$ with a memory footprint of $O(mn + mt + nt)$ memory, where m is the number of samples, n is the number of features and t is defined as $min(m, n)$ [79]. As a result, whereas the number of samples/features grows, the system becomes too resource-hungry to be adapted to a low-power platform. This is a common drawback shared among all these classical approaches, since they need to store all training examples for model computation, and this eventually limits the number of examples that can be considered for learning in a resource-limited embedded system.

Recently, a few works have investigated the use of deep neural networks, such as CNN [33] or combinations of CNN and Recurrent Neural Networks (RNN) [90]. Using complex topology of deep networks, these approaches can recognize up to 50 gestures with accuracy values ($>$ 80%), which is suitable for the design of a real time controller. Unfortunately, the training of deep neural networks requires a huge amount of data to converge and it requires dedicated GPU servers, resulting even more computationally demanding w.r.t. conventional approaches. Other techniques rely on linear and non-linear modeling of the arm movements, such as the work presented in [91] or the study proposed in [92], where EMG signals are decomposed to extract the neural information and classified with an HMM algorithm. In this vein, solutions based on blind separation of the motoneuron activation [32, 93], which aim for detecting which muscular fibers are actuated during the execution of a given movement, are rapidly gaining

Table 3.1: Comparison between SoA EMG embedded pattern recognition systems and our platform. Systems in [1] and [2] run in computationally hungry platforms and are not suitable for long-term operation, and, although the systems presented in [3] and [4] have a comparable energy/classification as this work, our system is capable of providing more than 2.7x improvement in battery life.

	Platform Architecture	ISA	Technology [nm]	Max Frequency [MHz]	Algorithm	Online learning	#Channles	#Gestures	E[mJ]/ class	Battery [h]
Benatti [3] :	1-core	Cortex M4	90	168	SVM	no	8	7	0.0891	9.97
Liu [1] :	1-core	Cortex A8	65	720	Muscoskeletal Model	no	8	-	1100.0000	0.27
Zhang [2] :	1-core	Cortex A8	65	720	LDA	yes	4	3	1100.0000	0.27
Gentile [88] :	1-core	Cortex M4	90	168	Threshold	no	1	1	1.0660	11.38
Liu X [89] :	1-core	Cortex M4	90	80	ANN	no	4	10	25.1500	2.94
Rossi [4] :	1-core	Cortex M4	90	168	HMM/SVM	no	4	6	0.0891	9.97
Falih [31] :	4-core	Cortex A53	65	1200	Naive Bayes	no	8	4	-	-
Benatti17 [39] :	4-cores	OR32Xpulp	130	40	SVM	no	3	3	2.8320	10.00
This work :	**8-cores**	**RV32IMFC Xpulp**	**40**	**500**	**HDC**	**Yes**	**8**	**11**	**0.0832**	**28.46**

ground. Such approaches rely on backpropagation techniques for the training and, moreover, they are based on high performance setup (i.e. >64 channels with bandwidth of several *kHz*). The lesson learned from this overview shows that the implementation of learning algorithms for pattern recognition is still an open challenge for EMG-based controllers. Luckily, in literature there are some attempts to move pattern recognition design gesture recognition systems toward a wearable form factor, complying with strict energy constraints.

Embedded implementation of computationally demanding algorithms requires a careful design of the system architecture and the digital computational platform. The most widely used computing platform is the ARM Cortex-M4 [94], a single core RISC-based processor designed for embedded applications. Thanks to the DSP extension of the ISA and Floating Point (FP) support, it represents a good trade off between computational capabilities and energy efficiency. It is widely used in EMG processing, for instance in [3] where a complete gesture recognition system on a ARM cortex-M4 microcontroller is implemented with a SVM used to recognize up to 7 hand gestures with a 8 channel EMG setup. Other works based on this architecture are presented in [88] and in [89] where an ANN with one hidden layer is implemented on a 4 EMG channel setup and it is capable to recognize up to 10 gestures with accuracy higher than 80%. Other works rely on more powerful platforms, to implement more complex models, like the one presented in [1], which leverages an ARM Cortex-A8 processor to execute the EMG gesture recognition using non linear modeling approach. However such high-end systems based on OMAP [95] feature a power consumption significantly higher than ones traditionally exploited for embedded platforms, which is not suitable for a wearable solution with reasonable battery lifetime. Finally, the work presented in [31] leverages a power-hungry Raspberry Pi board (up to 2.3 W of power consumption) with a 4-cores Cortex A53 where a Naive Bayes classifier is implemented to recognize up to 5 EMG gestures for the control of a wheelchair.

All the aforesaid works present the implementation of the classification stage of the algorithm, while the learning is done off-line on a bench-top PC. A first attempt to implement the learning on the embedded platform was tested in [2] where authors implemented the LDA learning on a ARM Cortex-A8 processor. Such a solution allows minimal learning capability but the setup is based on 4 sensors and can recognize only 3 gestures. The approaches presented in this chapter for learning and classification combine the low-power and high-performance capabilities of a PULP programmable platform with a novel brain-inspired algorithm [96] that is extremely robust against low signal-to-noise ratio (SNR) and large variability in both data and computing platform. Computational complexity of the proposed HD computing approach scales linearly with the number of electrodes [97] and maintains its accuracy with various types of biosignal acquisitions, while the PULP platform is highly optimized for parallel applications that require extreme energy efficiency.

Table 3.1 provides a quantitative summary of the state of the art of EMG-based pattern recognition embedded systems, including a comparison of energy per classification and battery duration (assuming a 100mAh battery). Systems in [1] and [2] are based on a high-performance ARM Cortex-A8 processor, and although they provide high computational power, they are not suitable for long-term operation due to their high power consumption. The systems presented in [3] and [4] have the lowest energy/classification ratio, which is comparable to our solution, but they lack battery life due to the high duty cycle needed. The system presented in this chapter leverages an output classification latency of 8ms, largely compliant with real-time requirements, and as a consequence, it provides more than 2.7x improvement in battery life compared to the SoA systems.

3.2.2 Support Vector Machine

The SoA algorithm for hand gesture recognition is represented by the SVM. Among pattern recognition algorithms, SVM has the main advantages of being theoretically robust and efficiently implementable [98]. SVM is a supervised learning algorithm belonging to the framework of statistical learning and its goal is to find the optimal separation hyperplane between 2 classes of vectors. This decision boundary, defined during the training stage, is obtained by a subset of vectors of the input space, named Support Vectors (SVs). In the case of a highly non-linear decision boundary, a kernel function maps data into a higher-dimensional space. The length of the model is variable and depends on multiple factors such as the nature of the data, the quality of the training set and the pre-processing capabilities. This model allows to classify unseen samples, computing the distance between the new instance and the decision boundary

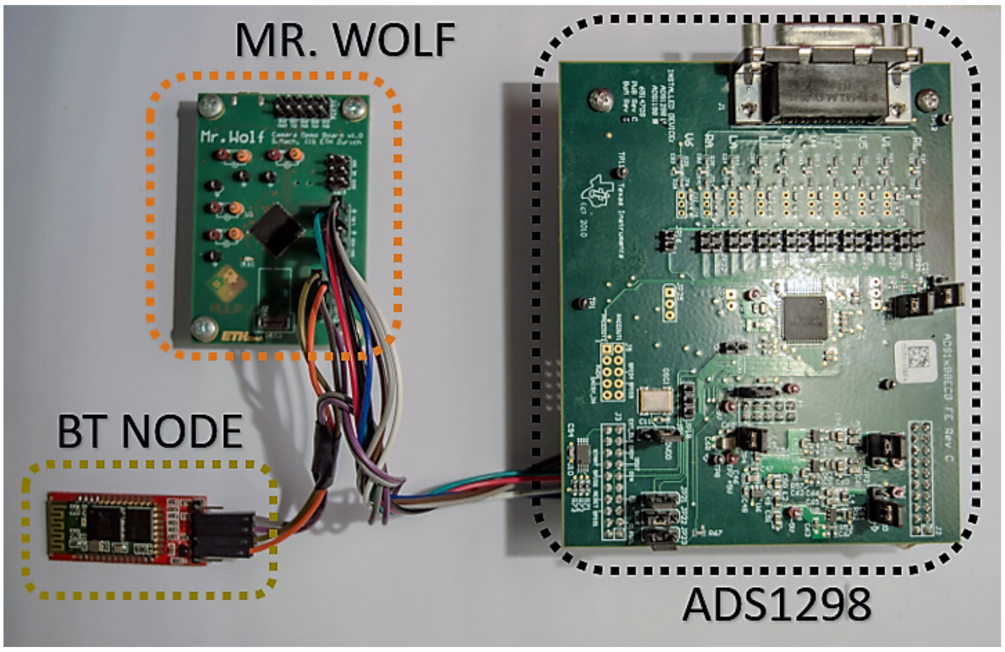


Figure 3.1: Picture of the tested system. EMG data are acquired with an external ADC (ADS1298), connected to the wolf chip. Output and communication are managed via a BT node.

through the formula:

$$f(\mathbf{x}) = \sum_{i=1}^{N_{SV}} y_i \alpha_i K\langle \mathbf{x}, \mathbf{s}_i \rangle - \rho \quad \begin{cases} f(\mathbf{x}) > 0, \mathbf{x} \in Cl_1 \\ f(\mathbf{x}) < 0, \mathbf{x} \in Cl_2 \end{cases} \tag{3.1}$$

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

$$K\langle \mathbf{x}, \mathbf{s}_i \rangle = \exp\left(-\frac{\|\mathbf{x} - \mathbf{s}_i\|}{2\sigma^2} \right) \tag{3.2}$$

The parameters of the classifier are tuned calculating a ROC curve [99], 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. Moreover, *sensitivity* and *specificity* define the quality of the classification for model validation:

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.3}$$

$$Specificity = \frac{TN}{FP + TN} \tag{3.4}$$

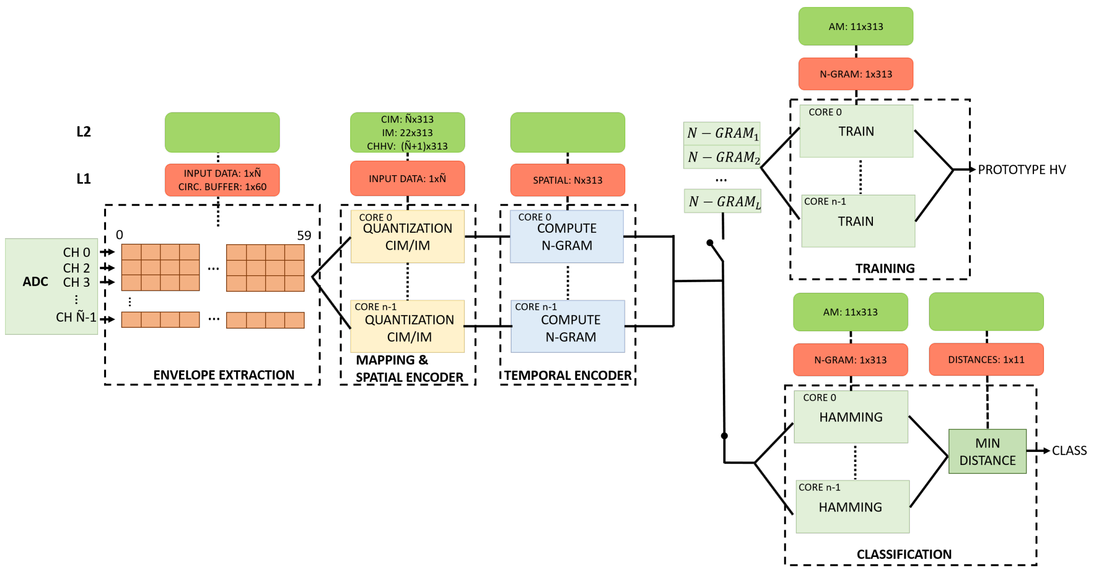


Figure 3.2: Implementation on PULP platform of the processing chain. The first three kernels are in common for both training and classification phase. The open/closed switches are there just for a visual idea and the meaning is that once the model is created during the training phase, we will not use that kernel anymore and we pass to the classification kernel.

where FN, TP, FP and TN are respectively False Negative, True Positive, False Positive and True Negative.

3.2.3 Hyperdimensional Computing

Brain-inspired HD computing explores the emulation of cognition by computing with points in a HD space, that is, with hypervectors, as an alternative to computing with numbers [96]. Hypervectors are d-dimensional holographic (pseudo)random vectors with independent and identically distributed (i.i.d.) components. When the dimensionality is in the thousands, e.g. $d \geq 1000$, there exist a huge number of nearly orthogonal hypervectors [100]. In HD computing we combine such hypervectors into new hypervectors using well-defined vector space operations, defined such that the resulting hypervector is unique with fixed-width.

To ease hardware implementation, dense binary hypervectors are considered, which are initially taken from $\{0, 1\}^d$ resulting in equal number of randomly placed 0s and 1s. HD Computing supports a full algebra to manipulate these seed hypervectors by using multiplication, addition, and permutation (MAP) operations. When using dense binary codes, MAP operations can be simply implemented by the componentwise XOR ($\oplus$) as multiplication, the componentwise majority function ([+]) as addition, and one-bit circular rotation (ρ) as permutation. The addition produces a hypervector *similar* to the input hypervectors and it can represent sets, while multiplication generates a *dissimilar* hypervector and is used to bind hypervectors. The permutation also produces a dissimilar pseudo-orthogonal hypervector, which is used to create hypervectors representing sequences of hypervectors.

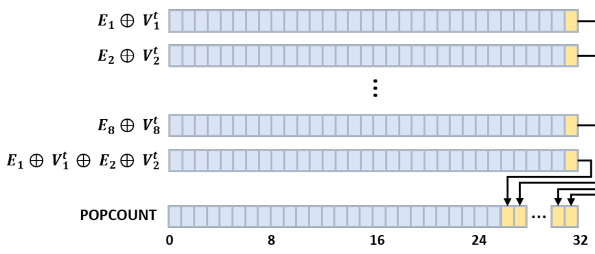
Based on the use of these MAP operations, an encoder can be designed for various tasks, e.g., EMG [78, 101, 102], EEG [103, 104], ECoG [105], ExG [97], or in general pattern processing [106]. The encoder generates a hypervector representing the event of interest that is then fed into an associative memory (AM) for training and inference. During training, the output hypervector of the encoder is added in the AM as a *learned* pattern. This allows continue updates into AM without requiring to store the raw data or features from the previous examples. During inference, the output of the encoder is compared with the learned patterns. Comparison is based on a distance metric over the vector space. The AM uses Hamming distance, defined as the number of different components of two binary hypervectors.

The encoding approach developed in this work starts with the mapping of the features. To map the features into the HD space we utilize item memory (IM) and continuous item memory (CIM) [101] matrices. The IM is composed of random orthogonal ($\perp$) hypervectors (i.e., $E_1 \perp E_2 ... \perp E_i$), each one associated to the i-th input channel. The CIM contains a set of orthogonal endpoint hypervectors, mapped on the discretized values of the input channels. For instance, if input values are discretized in K levels, we will have K hypervectors ($V_1..V_K$) where V_1 and V_K are associated respectively to the minimum and maximum input values. The hypervectors in CIM are generated by a linear interpolation between the two orthogonal endpoints [101]. The IM and CIM are initialized one time and kept constant during the training and the classification. Once the seed hypervectors are generated from IM and CIM, they are combined by the MAP encoders to represent the event of interest, e.g., a gesture. The first encoding is obtained with a componentwise XOR between E and V resulting, at instant t:

$$S^t = [(E_1 \oplus V^t_{l(1)}) + ... + (E_i \oplus V^t_{l(i)})]. \tag{3.5}$$

where E_i is the IM vector corresponding to the i-th input channel and $V_{l(i)}$ is the CIM vector corresponding to the $l(i)$-th discretized level of the input sample of channel i. Assuming that, at time 0, channel 1 acquires a new sample and, after the envelope extraction, a signal of amplitude equal to 21 is produced, we can bind this information by $E_1 \oplus V^0_{21}$. Then, the envelope extraction of channel 2 produces a signal with a smaller amplitude, for instance around 7, thus $E_2 \oplus V^0_7$. This is done for all the channels and the related amplitude values, resulting in $S^0 = [(E_1 \oplus V^0_{21}) + (E_2 \oplus V^0_7) + ... + (E_8 \oplus V^0_{15})]$ at instant 0.

In this way, the encoder captures the spatial information of a gesture, and can achieve slightly higher accuracy. However, if input data need a temporal component, the temporal encoder extracts information by considering N consecutive hypervectors, from instant t to $t+N-1$. The temporal encoder employs permutation and multiplication to capture order of hypervectors generated by the spatial encoder. Thus, N spatial hypervectors form an N-gram hypervector (T),



```
#pragma omp parallel num_threads(CORE)
{
#pragma omp master
{
        QUANTIZATION;
}
#pragma omp barrier
#pragma omp for
        for(i = 0; i < 313; i++){

        for(j = 0; j < CHANNELS + 1; j++){

            p.extructu      //bit 31
            p.insert        //bit 31
            ...
            p.extructu      //bit 0
            p.insert        //bit 0

        }
        p.cnt
    }
}
```

Figure 3.3: Illustration of how built-ins (p.extract, p.insert, p.cnt) used in the spatial encoder (left). A code snippet to show the usage of builtins and parallelism (OpenMP directives) in the processing chain (right).

defined as:

$$T = S^t \oplus \rho S^{t+1} \oplus \rho^2 S^{t+2} \oplus ... \oplus \rho^{N-1} S^{t+N-1} \tag{3.6}$$

where ρ^k means applying permutation k times (i.e., k-bit rotation of hypervector). The spatial and temporal encoders are common to both training and classification.

The N parameter used in the algorithm is kept constant (i.e. equal to 1) during both training and testing. During the training, an N-gram hypervector is generated for each trial and added as a *prototype* hypervector related to the class of the trial. The associative memory (AM) contains the same number of prototypes as the number of classes. During the classification, an N-gram is produced from unseen gesture that we call it *query* hypervector. In the AM, the Hamming distances are computed between the query hypervectors and prototypes hypervectors (as learned patterns) to find the label associated to the minimum distance.

3.2.3.1 Implementation on the PULP platform

HD Computing benefits of a computationally efficient learning algorithm, suitable for resource-constrained platforms. In this paragraph, the implementation of the online and on-chip training is described, giving details on the optimization and demonstrating the feasibility of this approach in fully-embedded energy efficient devices. Before starting the online on-chip training, the algorithm creates the matrices (*IM* and *CIM*) required for mapping the samples in the high dimensional space. They should be calculated once with parameters chosen for set-up (number of electrodes and quantization levels), at the beginning of the training phase, and they remain constant for the entire duration of the session.

The random orthogonal binary hypervectors are created through a random generation of 32-bit unsigned integer values. For instance, in an eight channel setup IM requires eight orthogonal hypervectors, which are generated through the *rand()* function (included in the GCC library) as 313 random 32-bit unsigned integer values, corresponding to a 10’000-D binary hypervectors ($313 \times 32 \approx 10'000$). The 10’000 dimension has been chosen to define a very high dimensional space able to demonstrate important properties of hyperdimensional representation [107]. Thus, the binary hypervectors are compacted into 313 32-bit unsigned integer variables (i.e. each bit represent one element of the hypervector) to drastically reduce the memory requirements of the application and, hence, the memory accesses.

Similarly, the CIM is computed considering the quantization levels required to represent the input data (i.e. 22 levels), and mapping these levels in an orthogonal hyperspace. The number of discrete levels for quantization has to be carefully selected, depending on the signal we are working with. After an exploration of the data acquired from the 10 subjects involved in the experiment and referring to [101, 102], we set a linear quantization on 22 levels, enough to approximate appropriately the input signals. A finer grain (i.e. more discrete levels) requires a higher number of hypervectors stored in memory (CIM matrix) with no relevant gain in performance, while a lower number of quantization levels affects the classification accuracy.

3.2.3.2 Implementation and Optimization on Mr.Wolf

Aggressive code optimizations can be achieved by compacting the hypervectors in 32-bit integers, leveraging simple operations such as componentwise majority, XOR and shift. The representation of the elements of the hypervectors using unsigned integer variables requires several bitwise operations (i.e. read/insert from/into a 32-bit word) and an operation to count the number of bits set to 1 in a 32-bit word (the so-called popcount). The Ri5cy augmented RISC-V ISA [102] includes several bit manipulation instructions (builtins) to perform this operation in 1 clock cycle, leading to an aggressive performance optimization. The builtins used in this application are *p.extractu, p.insert and p.cnt*. The formers two, *p.extractu and p.insert*, are used respectively to read and set the value assumed by a given bit in an unsigned 32-bit register, while the latter *p.cnt* returns the number of 1s set in a word. These builtins are used in the spatial encoder to calculate the componentwise majority after binding each channel to its signal level in the HD space and in the learning part to bound all the hypervectors associated to the training samples to create the *prototype* hypervector. Finally, the popcount is used to compute the Hamming distances.

In Fig. 3.3 an example of the use of the builtins is shown. The *i* components of the hypervectors need to be extracted (i.e., bit-by-bit) to perform the componentwise majority operation and to count the number of bits set to 1 for the majority voting. To perform the majority voting, the number of channels must be odd. Hence, if the number of channels (*i*) is even, to avoid randomness in the case of the same number of 0's and 1's, one reproducible but random hypervector is generated through a componentwise XOR between two bound hypervectors (i.e. the first and the second). For instance, in this case study, with 8 input channels, we use nine bound hypervectors to perform the majority voting. After that, the popcount (p.cnt) is used to evaluate if the number of 0's is higher with respect to the number of bits set to 1. If there are more 1's than 0's, the related bit is set to 1 in the spatial hypervector.

3.2.3.3 Parallelization and Memory Requirements

As shown in Fig. 3.2, the training and the testing phases have several functions in common. In particular, the feature extraction and the kernels used to map the features into the HD space are the same for both phases. When a new sample acquired by the ADC is available, the envelope of the signals is extracted, using the RMS upon a circular buffer of dimension *N* equal to 60. Performing several simulations, changing the dimension of the buffer from 30 to 150, it is possible to notice that between 60 and 150 the loss in accuracy is negligible (~1%), while using a smaller length leads to a more significant loss (~4% with a buffer size of 30 samples). The feature vector is composed of the envelopes extracted from the signals of each acquisition channel. After the feature extraction, the HD computing algorithm begins. The first kernel maps the features to the HD space, and performs the spatial encoding among all the acquisition channels. In this work, eight acquisition channels are considered. To map the acquired samples in the HD space, we use the CIM matrix, discretizing the samples in 22 linearly distributed levels. Then, the resulting hypervectors are combined with the one contained in IM related to each channel using the MAP operations.

In the MAP and SPATIAL ENCODER kernel, the parallelization is performed at data level, where the workload is equally distributed among the cores of the cluster. Hence, each core performs the encoding operations on a portion of the hypervector. The cores execute the componentwise XOR operation between CIM and IM and the componentwise majority to create the spatial hypervector. Fig. 3.3 (right) shows a code snippet of this kernel with the usage of the OpenMP directives for the parallelization. The spatial hypervector (1x313 32-bit integer array) is stored in the low latency access L1 memory and requires 2kB of memory. Due to the size of the CIM and IM matrices, respectively 22x313 (27 kB) and 8x313 (5 kB), they are stored in the L2 memory. The latency of the accesses to this memory is masked implementing a double buffering policy in which data are transferred from L2 memory to L1 memory through the DMA, while the cores are processing data already available.

If the temporal information contained in the signals is required (N-grams greater than one), the hypervectors in output to the SPATIAL ENCODER goes in the TEMPORAL ENCODER. In this kernel, a sequence of N spatial hypervectors are combined through a componentwise XOR operation after shifting them by one position as permutation. The vector in output to this kernel is the N-gram hypervector (it requires 2 kB, stored the in L1 memory), and serves as input for the AM kernel. During the training phase, the prototype hypervector is computed and stored in the L2 memory. The amount of memory required to store the AM matrix depends on the number of gestures we are considering. Storing a single prototype hypervector requires around 2kB, scaling linearly with the number of gestures. The IM, CIM and AM matrices are computed at runtime and stored in L2 memory for Mr. Wolf, while for the STM32F4-Discovery board they are stored in the FLASH memory.

3.2.4 Experimental Results

Evaluating the tradeoff between computational complexity and accuracy is one of the key elements in the selection of a ML algorithm for resource-constrained systems, especially when dealing with online learning capabilities. This section compares both training and classification performance of the proposed HD computing algorithm against the most widely used algorithm for gesture recognition (i.e. SVM), and their optimized implementation on Mr. Wolf and on a commercial ultra-low-power MCU (STM32F407), used as a reference for comparison. Ten able-bodied subjects (aged 26-42) without previous history of neurological or muscular disorders were involved in the experiment. All participants provided written consent to participate in the experiments. Initially, the system requires the subject to perform a given gesture and collect the data needed for the HD training.

The data were acquired with the SoA ADS1298 in the 8 channels fully-differential configuration. EMG passive wet Ag-AgCl electrodes (25mm diameter) were placed around the forearm at the same distance from each other with the arm facing downwards. Inter-electrode distance (i.e. distance between positive and negative channels) was set to 10mm. The gesture set was: *open hand, power grip, index pointed, 2-fingers pinch, wrist supination, wrist pronation, number two, number three, number four* and *rest position*, in the order shown in Fig. 3.4. During the acquisition, the subjects repeated each movement 6 times, holding the position with medium intensity of muscular contraction for 5 seconds and separating gestures with 5 second of rest position to avoid muscular fatigue. 25% of the samples for each acquired gesture were used for the training, while the whole dataset is used for testing. This approach has results very similar to the traditional procedure used in gesture recognition applications, where the data used for training and testing are separated by trials (i.e. some trials used for training and the others for testing). The two approaches were tested offline, obtaining an accuracy difference below 1%. This is due to the fact that gestures are basically ”static”, i.e. the contraction level is maintained almost constant all over the duration of the trial (excluding the transients), resulting in similar contraction patterns for all the trials of a given gesture.

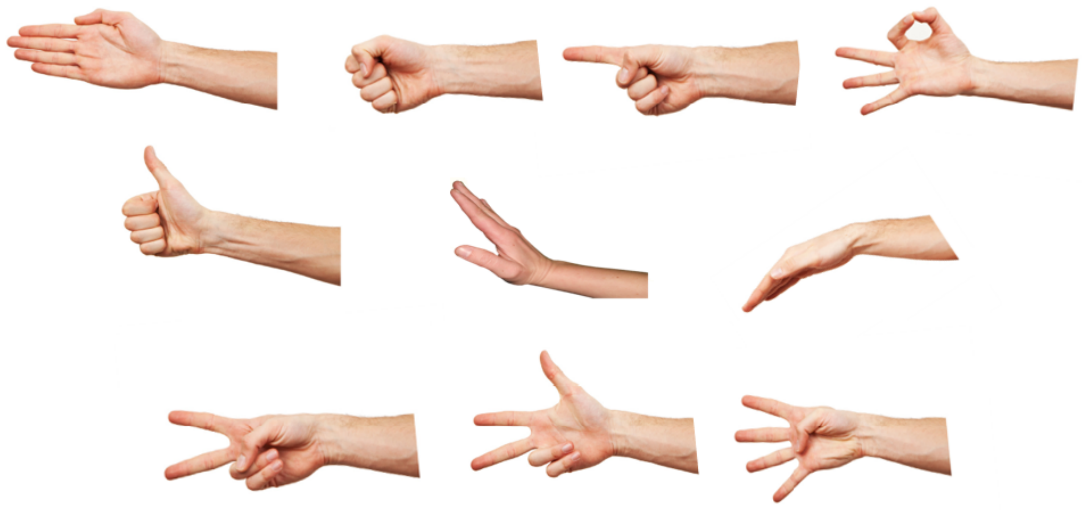


Figure 3.4: Gestures used for the testing. Open hand, fist, index, 2-fingers pinch, rest position.

3.2.4.1 Comparison of the HD computing with SVM on ARM Cortex M4

A comparison between classification accuracy of HD computing and the state-of-the-art SVM [16] is shown, measuring performance and power consumption of executing specialized serial versions of these two algorithms on a commercial embedded ARM Cortex M4, featuring the most commonly used ISA in the deeply embedded domain.

As the first step, the two algorithms are implemented and validated on MATLAB to establish a golden model to follow. An identical setup for both algorithms as presented in [101]; the model training is done per subject and off-line using 25% of the dataset, while the entire dataset is used for testing. The mean classification accuracy of gestures belonging to the five subjects is 89.6% with SVM, and 92.4% with the HD classifier. More importantly, the HD classifier exhibits a graceful degradation with lower dimensionality, or faulty components, allowing a trade-off between the application's accuracy and the available hardware resources in a platform [101]. This graceful degradation capability can be exploited by reducing the dimensionality of hypervectors that eases the execution on the commercial ARM Cortex M4. To do so, simulations are performed by reducing the dimensionality from 10,000 to 100. The HD classifier closely maintains its accuracy when its dimensionality is reduced from 10,000 to 200, but beyond this point the accuracy is dropped significantly. Therefore, for this experiment, the dimension is fixed to 200-D showing a mean accuracy of 90.7%, slightly higher than the SVM. This tuning allows compacting a hypervector to seven unsigned integers, and linearly reduces the number of operations of the HD classifier with no significant impact on its accuracy (i.e., iso-accuracy with SVM).

On the other hand, the SVM does not support such a flexibility. A trained model of SVM is composed by a number of support vectors (SVs). This parameter is not determined *a priori*, and

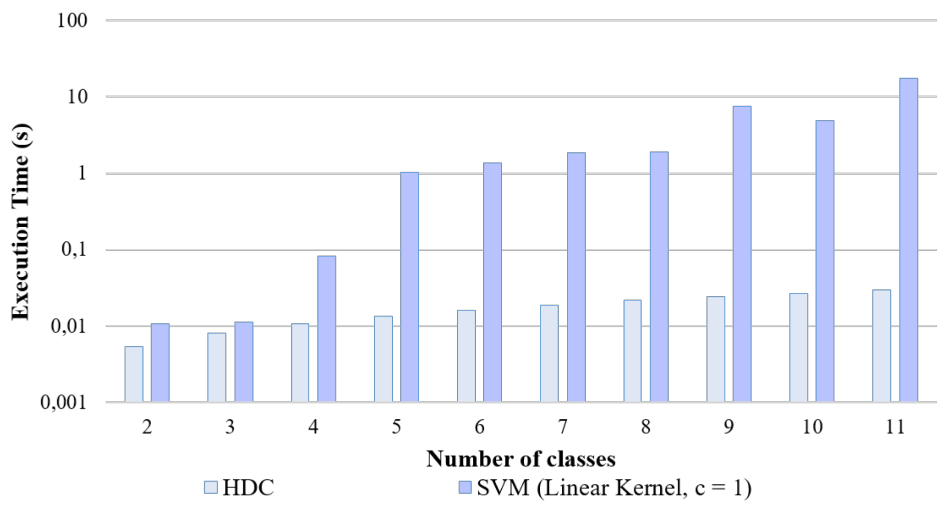


Figure 3.5: Estimation of the execution time for the training of SVM and HD Computing increasing the number of gestures (from 1 to 11). Results are presented in logarithmic scale.

Table 3.2: Comparison of HD computing (200-D) versus SVM at iso-accuracy on ARM Cortex M4. The results refer to a 10 ms detection latency.

	ARM Cortex M4	
Kernel	Cycles(k)	Accuracy(%)
`HD COMPUTING`	12.35	90.70
`SVM`	25.10	89.60

can vary due to several factors such as the scaling of the data, the kernel function, and the level of tolerable miss-classification. All this variability requires time to find the best configuration that leads to the smallest number of SVs maintaining the highest accuracy. Hence, a different number of SVs and the dimension of input feature vectors (i.e., the number of channels) induce substantial differences in performance. For this exploration, the dimension of the SVs is fixed to four as the number of input channels, while the number of SVs varies significantly across the model of five subjects, and finally is chosen to be 55 as the smallest among the subjects. This is in sharp contrast to the HD classifier since there is no variability in its model size after choosing its parameters: the dimension of the hypervectors, the N-gram size, and the number of input channels.

Table 3.2 summarizes the performance and accuracy results derived from the serial execution of the two algorithms on the ARM Cortex M4. The CIM, IM, and AM matrices of the HD classifier, and the SVs and coefficients matrices of the SVM, as the trained models, are loaded into the ARM Cortex M4 for testing. For SVM, a fixed-point approach is used to avoid all the computation needed to be executed in the floating-point. It is already demonstrated [19] that

Table 3.3: Detailed power (P) comparison of HD algorithm on the ARM Cortex M4 and PULPv3 based on number of cycles (CYC) and frequency (FREQ). The results refer to a 10 ms detection latency.

	CYC [k]	FREQ [MHz]	FLL P [mW]	SOC P [mW]	CLUSTER P [mW]	TOT. P [mW]	P BOOST [×]
HD COMPUTING							
ARM CORTEX M4@1.85V	439	43.90	-	20.83	N.A.	20.83	-
PULPv3 1 CORE@0.7V	533	53.30	1.45	0.87	1.90	4.22	4.9
PULPv3 4 CORES@0.7V	143	14.30	1.45	0.23	0.88	2.56	8.1
PULPv3 4 CORES@0.5V	143	14.30	1.45	0.23	0.42	2.10	9.9

this approach leads to best performance preserving the accuracy. As shown, the HD classifier achieves ≈2× faster execution and lower power at iso-accuracy compared to the SVM on the ARM Cortex M4. This is due to the fact that HD classifier mostly uses basic componentwise operations on the hypervectors.

3.2.4.2 HD computing on PULPv3 versus ARM Cortex M4

Table 3.3 shows the performance and power measurements of the HD computing on the PULPv3 prototype [108] in different operating conditions, and compares it with the ARM Cortex M4, benchmarked on an STM32F4-DISCOVERY board. In this experiment we use 10,000-D to retain to the best accuracy of 92.4%, and accordingly configure the clock frequency of the processors to achieve a detection latency of 10 ms [16, 109, 110]. The second column of Table 3.3 shows that with respect to the single-core PULPv3, the ARM Cortex M4 can operate at a lower frequency for the target detection latency, exploiting some optimized instructions that speed up the execution, namely *load and shift* and *load 32-bit immediate*. The key features of the PULPv3 SoC exploited in this work are performance-tunable near-threshold computing and parallelism. The 4.9× power gap between the ARM Cortex M4 and the single-core PULPv3 power at 0.7 V is partially given by the technology gap (i.e., 90 nm vs. 28 nm), but also by the cluster architecture and its implementation strategy optimized for energy-efficient operation [108]. Significant energy boost can be achieved through parallel computing over the 4 cores of the cluster. This allows to fully exploit the parallel compute power of the cluster and to reduce the operating frequency of the system by 3.72× (almost ideal speed-up over 4 cores) which saves significant power, leading to 8.1× power reduction with respect to the ARM Cortex M4, at the operating voltage of 0.7V. Finally, the process and temperature compensation capabilities of the SoC is exploited to enable aggressive voltage scaling, still reaching the target operating frequency of 14.3 MHz [108]. This key feature of the PULPv3 SoC allows to scale the voltage of the cluster down to 0.5 V, improving energy efficiency and leading to a power reduction of one order of magnitude with respect to the ARM Cortex M4.

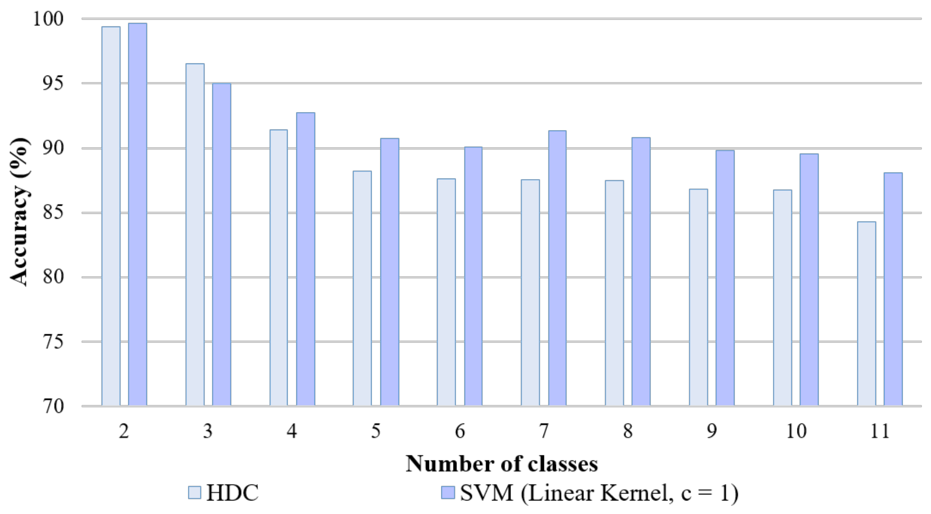


Figure 3.6: Average accuracy obtaining by SVM and HD computing, using the same data collected by 10 subjects, increasing the number of gestures (from 1 to 11).

It should be noted that the clock generation subsystem of PULPv3, composed of two frequency locked loops (FLL), is not optimized for low-power operation, featuring a reference frequency of 40 MHz and a power consumption of 1,45 mW. This block forms a bottleneck for energy efficiency at low voltage, dominating the overall power of the system. Replacing this block with a new generation FLL optimized for low-power [111] would reduce the clock generation power by 4× leading to a further 2× reduction of system power, and boosting energy efficiency by ≈20× with respect to the ARM Cortex M4.

3.2.4.3 Comparison between SVM and HD computing of Training Times and Classification Accuracy

One of the most important advantages of HD computing is the possibility to perform the training of the algorithm 'one-shot', paving the way to the real-time implementation and execution of the training phase. This section compares the computational load for the training of the SVM against HD and the accuracy in classification, starting with two gestures and scaling up to determine the computational cost of the two algorithms in different application scenarios (i.e. where a different number of recognized gestures are required). In this experiments, the SVM uses a linear kernel (c=1), with data normalized between 0 and 1. Such configuration has been selected after experimental evaluation, and represents the best trade-off between performance and computational complexity for this setup.

While it has been observed that, for some signals such as EEG, the temporal information is extremely important to achieve good classification accuracy, this is not the case for EMG signals [3, 101], whereas experiments are based on "static" gestures. Thus, the results in this section are

Table 3.4: HD Computing Execution times on the target architectures, with 10,000-D, N=1. (Cyc, su) stand for (cycles, speed-up). The total energy/class reported, is the result of the addition of the contribution of these functions without considering the energy during idle periods.

		ARM CORTEX M4			Mr. Wolf 1 core				Mr. Wolf 1 core built-ins				Mr. Wolf 8 cores built-ins			
	Kernel	**cyc(k)**[a]	**cyc(k)**[b]	**E(μJ)**[d]	**cyc(k)**[a]	**cyc(k)**[b]	**su**[c]	**E(μJ)**[e]	**cyc(k)**[a]	**cyc(k)**[b]	**su**[c]	**E(μJ)**[e]	**cyc(k)**[a]	**cyc(k)**[b]	**su**[c]	**E(μJ)**[e]
TRAIN	RMS	13.78	399.62	202.42	6.82	197.78	2.02	24.99	6.82	197.78	2.02	24.99	0.89	25.81	7.66	5.13
	MAP+ENCODERS	537.71	15’593.59	7’898.90	569.10	16’503.90	0.94	2’085.49	215.35	6’245.15	2.50	789.16	27.94	810.26	19.25	161.22
	TRAINING		41’305.84	20’923.37		25’620.47	1.61	3’237.50		16’696.69	2.47	2’109.85		2’136.18	19.34	425.05
	TOT TRAIN		57’299.05	29’024.70		42’322.15	1.59	5’347.99		23’139.60	2.47	2’924.01		2’972.25	19.33	591.41
TEST	RMS	13.78		6.98	6.82		2.02	0.86	6.82		2.02	0.86	0.89		7.66	0.17
	MAP+ENCODERS	537.71		272.37	569.10		0.95	71.91	215.35		2.50	27.21	27.94		19.24	5.55
	AM	70.65		35.78	68.59		1.03	8.66	24.19		2.92	3.05	7.23		10.06	1.43
	TOT TEST	622.15		315.14	644.48		0.97	81.44	246.37		2.53	31.13	36.06		17.25	7.17

[a] cycles per sample, [b] cycles per class, [c] speed-up wrt ARM Cortex M4,
[d] 168MHz@1.85V, [e] 100MHz@0.8V

presented considering N=1 for all subjects (the same for training and classification). Results are shown in Fig. 3.5. The graph is plotted in logarithmic scale for the sake of clarity. It is noteworthy that the training time of HD computing algorithm is deterministic and grows linearly when the number of classes increases. On the contrary, SVM training time is not deterministic (i.e. it depends on the convergence time of the algorithm) and, more importantly, it increases superlinearly with the number of gestures, resulting not affordable for embedded implementation if the number of gestures is higher than 5. Overall, the training time of the SVM is between 2 and 3 orders of magnitude higher than the HD Computing.

After the characterization of the training time of the SVM against the HD computing we have measured the accuracy of the classification obtained with the two classifiers. The average accuracy values are shown in Fig. 3.6. Results are comparable in terms of accuracy, showing that HD computing has a maximun 4% accuracy loss wrt SVM, resulting suitable for a hand gesture controller [16].

3.2.4.4 Evaluation of Execution Performance

In this section presents results of the HD computing implementation (both training and testing) on the proposed embedded system. Table 3.4 shows the execution times of the algorithm on Mr. Wolf incrementally exploiting the features of the architecture, namely the DSP extensions available on the core (Mr. Wolf 1-core built-ins), and the parallelism of the 8 cores available on the cluster (Mr. Wolf 8-cores built-ins). The implementation on an STM32F4-DISCOVERY board (ARM Cortex M4) is also shown in the table as reference. Results refer to hypervectors of 10’000-D, *N* equal to 1 (i.e. no temporal information), 11 classes (10 gestures and the rest position).

The *N* parameter is fixed to 1 to show the fastest case for training and classification, and also because the MAP+ENCODERS kernel increases linearly, increasing the temporal window length. This results also in a better generalization of the application, since the same level of temporal encoding is used for all the subjects. In more subject-specific applications (beyond

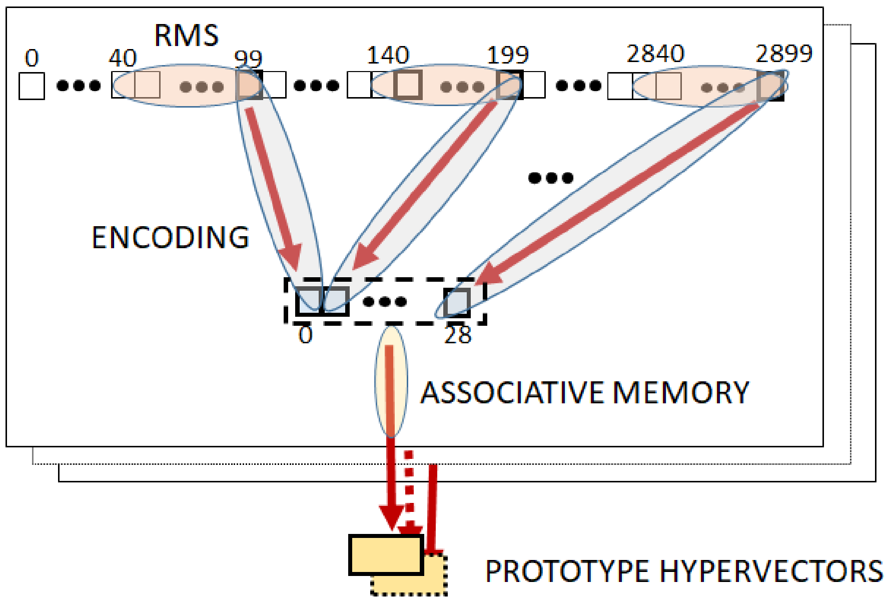


Figure 3.7: Sequence of the computational kernels of the HDC training. The first kernel is the RMS, computed on 60-samples windows, every 100ms. Output of the RMS is used by the encoding kernel to calculate the AM vectors. Once the 29 vectors of the associative memory are calculated, they are used to calculate the prototype hypervectors (1 for each class of the problem)

the scope of this work), it is possible to adapt the temporal encoders scaling the value of N, adding more temporal information, besides the RMS [101]. Considering the upper part of the table (TRAINING), the first column shows the cycles required for the execution of the RMS and MAP+ENCODERS for one sample (each 100ms). To better understand the part of the processing we are taking into account, Fig. 3.7 shows where the computational kernels are executed.

Theoretically, to collect the training set, there is the need of continuously extract the envelope from the signals. As already discussed, to reduce the number of samples used for the training and to cover the entire shape of the gesture, we use a downsampling factor of 100 (this factor can vary depending on the number of samples used for the training and the duration of the gesture). In this way, it is possible to compute the RMS only 1 time after the acquisition of the 100th sample (Fig. 3.8). In the RMS kernel we can see that just changing the architecture (passing from ARM Cortex M4 to 1-core Mr. Wolf) leads to an improvement of 2.0×. This improvement is mainly due to the hardware loops and the floating-point Fused Multiply and Accumulate (FMA) available in Mr. Wolf. In fact, in this part of the processing chain, we are considering floating point variables that are successively binarized into hypervector. This kernel is parallelized splitting the acquisition channels (i.e. 8, 16, 32, etc.) to calculate envelope values among the cores inside the cluster reaching a nearly ideal speed-up.

The MAP+ENCODERS kernel results slightly slower when executing on a single-core of

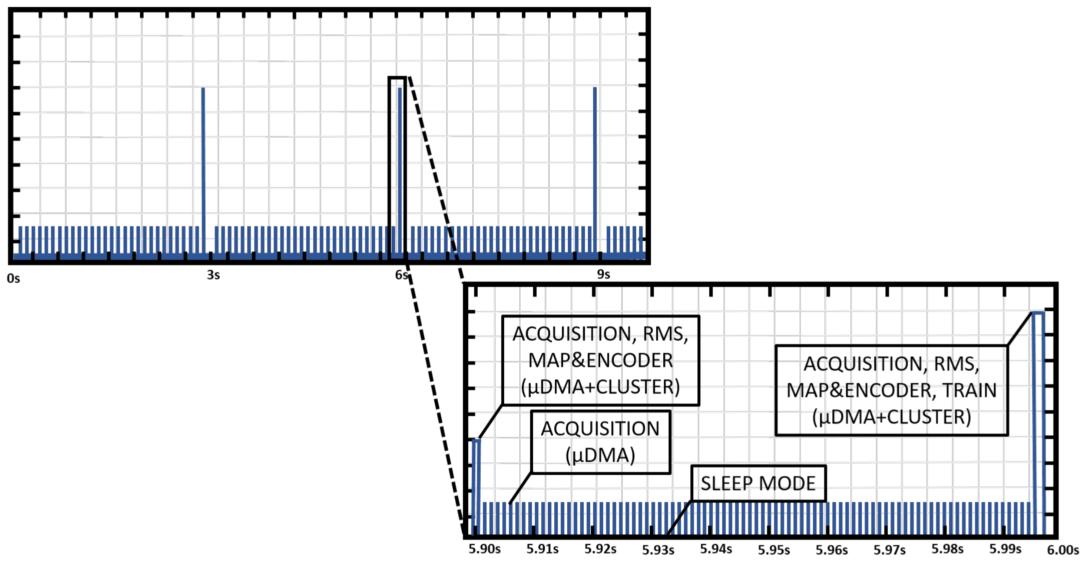


Figure 3.8: Power envelope of the training algorithm. EMG samples are acquired during *Acquisition* steps,whereas PULP cluster is turned off and uDMA manages data transfer from SPI to uDMA. Once the buffer for RMS computation is ready, the cluster is switched on and calculates RMS value and hypervectors for the encoder (*RMS MAP and ENCODING*).

Mr. Wolf, when no bitwise manipulation extensions are being used. This is due to some of the instructions available on the ARM Cortex M4 architecture (i.e. *load and shift* and *load 32-bit immediate*), useful to compute the majority function (based on population count operation), heavily relying on load-and-shift and 32-bit load immediate operations not available in the baseline RISC-V Instruction Set Architecture (ISA). However, this kernel can be highly optimized in Mr. Wolf taking advantage of the DSP extensions. Since the built-ins to extract/insert bits into a 32-bit word require as input a 5-bit immediate to indicate the position of the bit we are considering, we unrolled the inner loop including this function to remove the dependency with the loop index, fully exploiting the capabilities of these extensions, leading to an improvement of 2.5× over Cortex M4 (when considering a single core).

Furthermore, this kernel demonstrates almost ideal parallelism due to the lack of dependences between the different iterations of the inner loops. In fact parallelizing the execution of the code on 8 cores improves this gain up to 19.2× with respect to the ARM Cortex M4 execution time, showing a nearly ideal speed-up when comparing its execution time with respect to the execution on a single core (i.e. 7.7×). The last kernel takes all the N-grams related to the training samples and creates the prototype hypervectors for all classes. Built-ins can be effectively used also in this case to optimize the execution, since the majority operation is required also in this kernels, leading to 2.4× better performance (unrolling the loop as in the previous case) compared to ARM Cortex M4 and 19.3× when exploiting the parallel processing cluster.

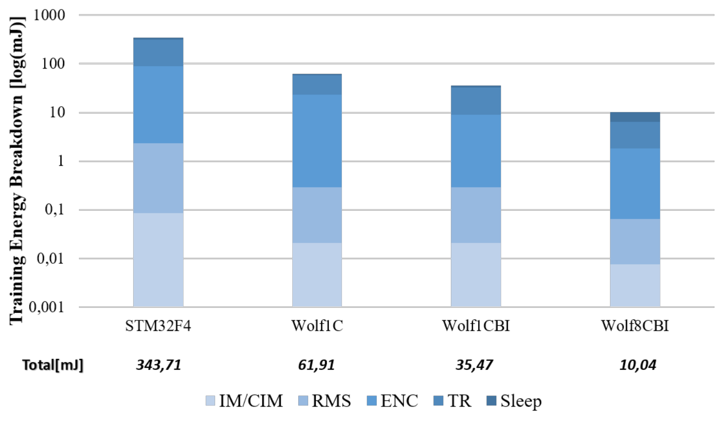
In the second part of Table 3.4 (TEST), results for the testing of the HD Computing algorithm are shown. The first two kernels (RMS and MAP+ENCODERS) have been already discussed in the description of the training results. The last kernel is the AM, where the *query* is classified in one of the possible classes (11 in this particular case). Exploiting the parallel execution leads to a saturation on the speed-up because of the small quantity of workload to distribute to the cores. In fact, the speed-up obtained using 8-cores Mr. Wolf is equal to 3.3× (10.0× wrt ARM Cortex M4). This small gain does not impact significantly on the overall speed-up (6.8× wrt 1-core Mr. Wolf with builtins and 17.3× wrt ARM Cortex M4) as the dominant part is the MAP+ENCODER kernel.

3.2.4.5 Evaluation of Energy Consumption

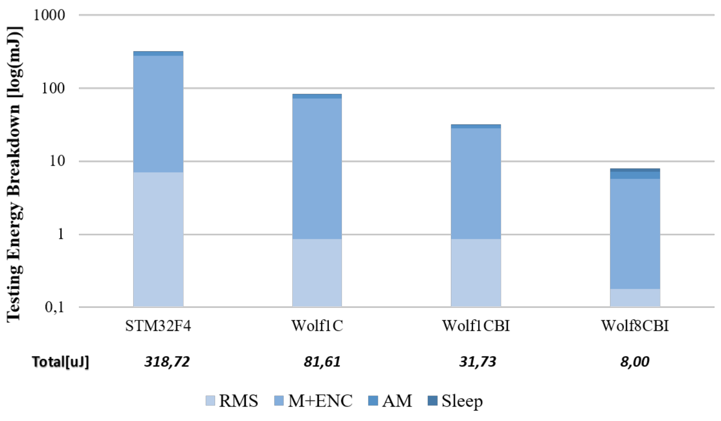
This section presents a comparison in term of energy consumption between the target architectures. The commercial ARM Cortex M4 MCU works at an operating frequency of 168MHz and 1.85V, while we set the operating frequency of Mr. Wolf architecture to 100MHz (the maximum is 450MHz) at 0.8V, the most efficient operating point. Each gesture is held for three seconds. EMG signals are acquired at 1kHz, hence a new sample is available each 1ms. To reduce the number of samples required to train the algorithm, while maintaining a clear idea of the shape of the gesture, we calculate the RMS value on 60 samples each 100ms (see Fig. 3.7). Hence, the acquired samples can be stored in memory using the μDMA keeping the PULP cluster switched off, and then, after the acquisition of the 100th sample, we can wake up the cluster and perform the RMS and the MAP+ENCODERS kernel creating a new hypervector.

As soon as all the hypervectors required by the training are generated, the prototype hypervector, which will be used for the classification, can be generated. When no elaboration and no acquisition are required, we can put the cluster and part of the SoC in deep sleep to further save energy. Table 3.4 also shows the energy consumption for each kernel of the training and the testing of the algorithm. The total energy consumption for the training of the algorithm on the ARM Cortex M4, including all eleven gestures, is equal to 29.02mJ, while the energy consumption on the single core Mr. Wolf is 5.35mJ (5.4 energy boost). This improvement mainly derives from the difference in technology (40nm for Mr. Wolf and 90nm for ARM Cortex M4) and the differences in the operating state. The ARM Cortex M4 operates at its maximum operating frequency (168MHz) at 1.85V, while for Mr. Wolf, which can work at a maximum frequency of 450MHz, finds its most efficient operating point at 100MHz. We can also lower the voltage operating in near-threshold (0.8V). The use of built-ins can improve this gap in energy and exploiting the parallel computing on eight cores, reaching an energy boost of 49.1×.

During the classification, the energy boosts are more significant because they represent the energy consumed by the MCU for most of the lifetime of the application. The boost achieved by changing the architecture (from the commercial MCU to the single core Mr. Wolf) leads to



(a)



(b)

Figure 3.9: Energy breakdown during training (a) and testing (b) using the HDC on the target platforms. The bars are plotted in log scale to better show the energy contribution of each function. The total energy consumption at the bottom of the figure is reported in linear scale. Working with 8-cores, Mr. Wolf achieves the lowest energy consumption, saving up to 34x energy with respect to the commercial STM32F4 MCU.

a 3.9× improvement in energy. Optimizing through the built-ins gives a further 10.1× improvement while splitting the workload among eight cores allows us to obtain up to 44.0× gain, with an energy consumption of 7.2μJ. Considering also the energy required by Mr. Wolf (8-cores) at deep sleep, the system can provide nearly 300h of autonomy. In this evaluation, we do not include the ADC power to better showcase the benefits of the architecture.

Fig. 3.9(a) and 3.9(b) show the energy breakdown during the training an testing. Both plots are represented in a logarithmic scale to better exhibit the energy contribution of each function at different orders of magnitude. In fact, this is already denoting the benefits of using Mr. Wolf as a processing architecture. During both phases, the energy consumption is the result of the contribution of the power modes. When processing is required, the MCUs will run

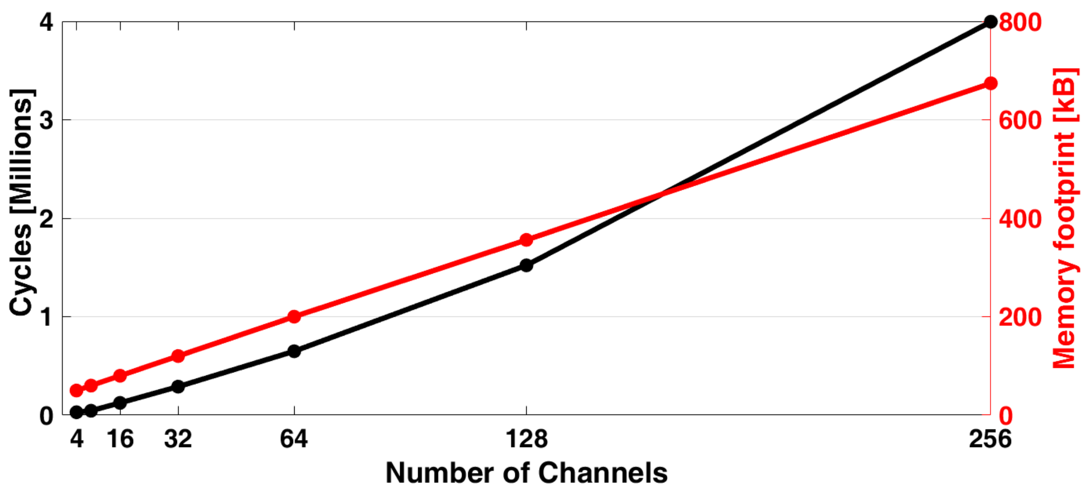


Figure 3.10: Performance and memory footprint of HD computing, increasing the number of channels. The results refer to 8 cores Mr. Wolf execution with built-ins.

at the maximum allowed core frequencies (Run mode), and when in idle, they will be put in deep-sleep.

During training, the contributions in energy for the execution of the kernel functions in nearly all architectures are similar, i.e., dominated by the ENC and TRAINING kernels. The case is different for Mr. Wolf (8-cores), where the energy required during the active time becomes comparable with the energy employed in deep sleep mode, thus, approaching the lower bounds of the energy consumption. This is possible by taking advantage of the parallelization of the code and the technology improvements previously presented. The same trend is noticeable during the testing (i.e., real-time classification), where the energy of most computationally expensive function, MAP+ENCODERS (M+ENC) kernel, becomes more comparable with the deep-sleep energy when moving towards the architectures with higher core count. As a result, Mr. Wolf (8-cores) works with an average power consumption of 0.82mW, demonstrating that complex signal processing and ML can be executed at an extremely low power budget.

3.2.5 Scalability

The results presented in Section 3.2.4 focus on the use case addressed in this work for EMG hand gesture recognition with 8 channels and an N-gram size of one. This HD computing framework can handle more complex tasks (i.e. EEG processing) where a higher number of channels for a larger coverage and larger N-gram size for a wider temporal window are required[102]. To demonstrate the capabilities of this framework to handle other type of applications, the scalability is assessed by increasing the number of channels from 4 to 256 channels.

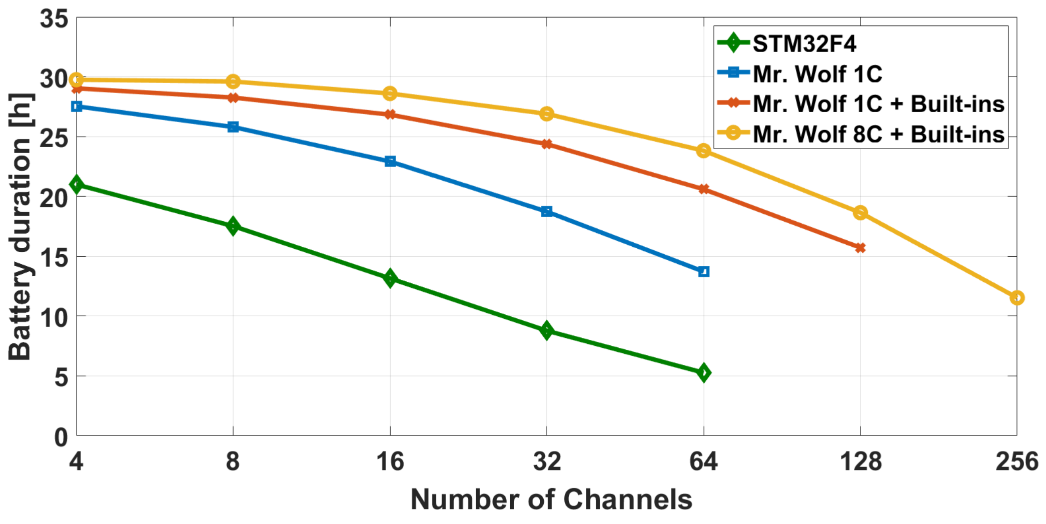


Figure 3.11: Battery duration of the target applications including the static power of the ADC with an increasing number of channels. With a fixed latency of 40ms per classification, Mr. Wolf 8-cores, even with a single core, can provide the same performance than the commercial STM32 (up to 64 channels).

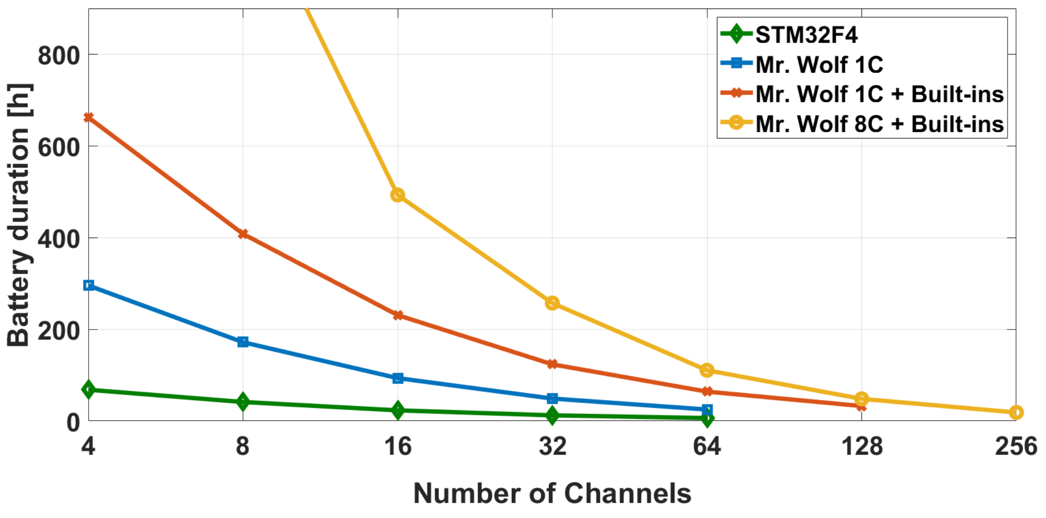


Figure 3.12: Battery duration of the target applications with an increasing number of channels. This figure excludes the static power of the ADC to better showcase the power characteristics of the target architectures. The single core version of Mr. Wolf (without built-ins) can provide the same autonomy than the commercial ARM Cortex-M4 MCU (up to 64 channels). With built-ins, the same core offers more than one day of battery life (up to 128 channels), and the 8-core version, up to 18h (256 channels).

Increasing the number of channels affects both execution time and memory requirements. As shown in Fig. 3.2, the processing chain can be divided in two part. The first part includes the kernels ”channel dependent” (RMS and MAP+ENCODERS), for which the performance changes increasing the number of channels, while the second part includes the kernels ”channel independent” that have no impact on the performance scaling up the number of channels. Fig.

3.10 shows that the RMS and the MAP+ENCODERS kernels linearly increase the execution time scaling up the number of channels (from 4 to 256), so as the memory requirements for the correct execution of the processing chain. This can be extended for the testing part, as the number of cycles for the AM kernel (channel independent) decreases the impact on the overall performance as the number of channels increases. Similarly, for the training part, the number of cycles reported on the graph has to be multiplied by the number of samples required for the training of each class and the cycles required to create the prototype hypervector has to be added.

Furthermore, the impact on the battery life when scaling the number of channels for the training part is also explored. Fig. 3.11 shows the autonomy of the target architectures including the static power of the ADC and considering a battery of 100mAh. With a classification latency of 50ms, the STM32 and Mr. Wolf (1-core without built-ins) can only be scaled up to 64 channels because they are not able to meet this latency constraint. Nevertheless, Mr. Wolf denotes a more efficient operation by providing up to 3.9x more autonomy than the STM32. If we include the built-ins, the same core can now be scaled up 128 channels, providing more than 48h of continuous operation and demonstrating the benefits of the built-in instructions. Nonetheless, for the current application, Mr. Wolf (8-cores with built-ins) exceeds all other architectures (for every number of channels) mainly thanks to the distribution of the task into the different cores.

Another interesting fact comes from the line shapes, where all Mr. Wolf architectures will show a pseudo-constant average autonomy when increasing the number of channel up to 64 (with respect to four channels), with only 32% of autonomy degradation, while the commercial MCU will show peak degradation of 74%, i.e., a linear and steep decrease. This difference is due to the negligible power contribution of Mr. Wolf with respect to the ADC power. When scaling up the system with more channel, Mr. Wolf will still provide 19h to 12h of operation for 128 and 256 respectively, adequate for daily use.

Fig. 3.12 also presents the autonomy of the architectures without including the ADC contributions, which can be subject of further optimizations depending on the application. This representation also highlights the benefits of using the PULP architecture, where again, Mr. Wolf exceeds the commercial MCU, being able to provide from 1'400h to 19h of continuous operation with an increasing number of channels.

3.2.6 Discussion

In this chapter, a complete framework for EMG gesture recognition is presented. The solution includes a novel algorithm for fast online training, implemented on a programmable multi-core platform. Leveraging the hardware and software co-design, which ranges from algorithm to embedded optimization, to enable a simple and scalable solution for supervised pattern recognition. In fact, the supervised learning approach is widely used in pattern recognition applications, but it lacks versatility, due to heavy computational requirements of the training stage.

This solution has been tested on 10 subjects in a typical gesture recognition scenario. HD computing with online learning reaches 85% accuracy on the recognition of 11 hand gestures, which is aligned with the SoA. Furthermore, by virtue of the efficient Mr. Wolf multi-core processor, the energy budget required to run the learning part with 11 gestures is 10.04mJ, and 83.2μJ for one classification. The system works to an average power of 10.4mW in classification, ensuring around 29h of autonomy with a battery of 100mAh.

As future work, the system can be integrated in a miniaturized form factor and the power consumption further optimized. Finally, this solution can be integrated and tested for a higher number of channels, extending the HD computing based solutions for other biopotentials, such as EEG signals.

Chapter 4

EEG Applications for Brain Machine Interfaces

4.1 EEG Signals

4.1.1 Signal Description and Acquisition

The brain tissue is an agglomerate of cells (neurons) in which an electrical current flows generating the electrical activity that can be measured by EEG. A neuron is composed of a cell nucleus (i.e. cell body) connected to other neurons through dendrites and axons. The dendrite allows collecting and aggregate signals from other neurons, while the axon is used to send information to other neurons. The communication is done through neural impulses. The human brain contains strongly interconnected neurons (in the order of 10^{11}), and these specialized connections between neurons are called synapses. A cortical axon can be covered by a conspicuous number of synapses (between 1'000 and 100'000).

The structure of the neural cell is very simple, but the communication between multiple cells relies on complex models of ionic current generation and conduction. The neurons are excited by other neurons through action potentials (APs) and generate excitatory postsynaptic potentials (EPSPs) from the aptic dendritic branches. A current is generated by a potential difference caused by the depolarization of the zone between the dendritic membrane and the cell body. This current, also known as primary current, flows from the membrane of the cell body to the dendritic tree (the EPSPs). A secondary current, extracellular, is looped with the primary current for the principle of conservation of the electrical charges, generating the magnetic field that can be measured by electrodes placed on the scalp of the subject. Considering that the magnetic field generated by these currents has to pass through several layers (soft layer, skull, etc.), the signal is attenuated dramatically before being captured by the electrodes. For this reason, the signal that is acquired is the result of the superposition of the neural currents produced by thousands of synchronously activated neurons.

The acquisition of EEG data is performed through non-invasive approaches that substituted other more invasive techniques such as MEG, fMRI, ECoG. In fact, to acquire the EEG signal is sufficient to place electrodes on the scalp of the subject and to measure electrical potential differences between pairs of these electrodes. Nevertheless, the EEG signal is intrinsically noisy - due to the presence of other brain activities that may be unrelated to the potential we want to consider, and other external disturbances connected to the acquisition system (electrical noise, external interference, etc.). Moreover, we should consider other artifacts (movements, eye-blinks, etc.) that make the acquisition process not easy. The contact between the electrode and the skin itself represents a source of noise that degrades the EEG signal (i.e. contacts impedance). This interference can be minimized by removing superficial skin layers and applying a gel between the two surfaces. This procedure leads to a longer set-up time, which can result in being uncomfortable and tricky to perform. Hence, zero-preparation time solutions can be preferred, using dry electrodes, trying to minimize the quantity of high contact impedance derived from this choice. Adding a simple amplifier stage directly on the electrode can mitigate this side effect [112].

4.2 Automated Wide-Range Frequency Tagging Analysis in Embedded Neuromonitoring Systems

This chapter presents a framework for automated analysis of FT responses in EEG traces from FT stimulation protocols, with the specific purpose of application both in the standard frequency range (>6Hz) and in the low-frequency range (0.5-6Hz) of FT stimulation. The algorithm, using machine learning techniques, automatically removes the EEG segments affected by artifacts of various origin and detects the presence of the resonant frequency even at very low (< 1Hz) stimulation frequencies, for which detection is typically very challenging. The algorithm is tested on 4 subjects undergoing a visual stimulation with an on-off checkerboard pattern at 0.667Hz, 0.8Hz, 4Hz and 12.5Hz, showing that automated processing is able to detect the presence of resonance at all the target frequencies. Results are compared with the manual processing performed by an expert human. Furthermore, the implementation and the profiling of the algorithm on a programmable ultra low power multicore platform (PULP) is introduced, demonstrating that it is possible to design a fully wearable system for autonomous on line detection of resonant frequencies during *in vivo* tests, featuring a power consumption of 56mW leading to a battery-life of almost 24h.

Table 4.1: Comparison with state-of-the-art of Artifact Removal.

	Barbati [113]	Delorme [114]	Mantini [115]	Li [116]	Viola [117]	Joyce [118]	Okada [119]	Mognon [120]	**This Work**
STIMULI	-	visual	auditory, visual	visual	auditory, visual	-	visual	auditory, visual	visual
HUMAN INTERVENTION	✓	✓	✓	✓	✓	✓	✓	x	x
ICA	✓	✓	✓	✓	✓	✓	✓	✓	x
DOMAIN	time	time	time	space	space	both	both	both	both
EMBEDDED	x	x	x	x	x	x	x	x	✓
REAL-TIME	x	x	x	x	x	x	x	x	✓
NUMBER OF ELECTRODES	28	32	153	32	30-128	6	306	64	64

Table 4.2: Comparison with state-of-the-art of Frequency Tagging.

	Buiatti [51]	Ding [121]	Baldauf [122]	Kim [50]	Rossion [52]	Karbdebon [53]	de Heering [54]	**This Work**
STIMULI	auditory	auditory	visual	visual	visual	auditory	visual	visual
METHOD	FT	FT	FT	FT	FT	FT	FT	FT
FREQUENCY RANGE [Hz]	1.4-4.2	1-4	1-2	12.5-16.67	1.18-16.37	1.39-4.17	1.2-6	0.667-12.5
EMBEDDED	x	x	x	x	x	x	x	✓
NUMBER OF ELECTRODES	129	157	-	59	128	64	32	64

4.2.1 Related Work

Processing of EEG data is a complex and multiple steps process involving filtering, feature extraction, machine learning and classification techniques [123]. A general structure of BCI system is presented in [124], showing a series of functional processing blocks among which signal pre-processing, artifact removal, feature extraction that can be preceded by signal enhancement and accompanied by dimensionality reduction, feature selection, classification and post-processing. In case a direct real-time feedback is required, further processing steps are considered such as algorithms for actuator control, tuning for adaptive and smart feedback, etc. The complexity, the non-automatization and the number of processing steps are some of the barriers to the design and use of wearable EEG solutions in daily life [125].

It is convenient to tailor the choices of the algorithm for artifact removal, as well as those for the following steps, such as feature extraction and classification, to the target application. The attention is focused on experimental protocols based on FT designs, where the response to be detected is determined by the stimulation frequency. In this case, the influence of artifacts on the response depends on the stimulation frequency. Since most artifacts (blinks, eye movements, heart beats) are characterized by a broad frequency range mainly on frequencies lower than approximately 6Hz, the response to visual or auditory periodic stimulation at frequencies higher than 6 Hz (the standard frequency range of SSVEP) is easily detectable with very light artifact correction, and efficient techniques for on-line processing of SSVEP-based BCI already exist (e.g. [49]). Conversely, for stimulation frequencies in the low-frequency range (0.5-6Hz), accurate artifact correction/removal is fundamental for the correct identification of the frequency-tagged response, although artifact rejection thresholds are generally 2-3 times higher than for ERP designs [51]. Despite the recent rising of studies using FT designs in the low-frequency range [51] [121] [122], artifact correction/removal employed in these studies is usually the same used in ERP designs with lighter thresholds, and no tool specific for low-frequency FT design exists in the literature.

Several solutions have been proposed to perform automatic EEG artifact removal. For example, ICA, which is one of the most efficient methods for artifact identification, has been subject to several attempts of semi-automatization in time [113] [114] [115] and space domain [116] [117] or both domains [118] [119]. An interesting fully automatic approach is presented by Mognon et al. [120], based on the automatic adjustments of the algorithm's parameters to the data used for the spatial and temporal feature extraction for IC classification. However, this method still requires to process data in two domains, time and space, and fuse results, and is uniquely based on the computationally intensive ICA decomposition. It therefore presents challenges for a resource constrained device and it can only be performed off-line. Table 4.4 shows a summary of comparison between the state of the art of artifact removal solutions. Table 4.2 present the state of the art in FT solutions. The comparison covers the most relevant characteristics mentioned in the references.

Concerning the hardware requirements, most of the previously mentioned algorithms are meant to run on desktop machines or servers, eventually equipped with real-time monitoring tools which synchronize through a USB cable with the helmet for EEG tracks visualization, while data analysis is often performed off-line using EEG feature extraction and interpretation toolboxes such as EEGLAB [126] or OpenViBE [127]. While most traditional EEG monitoring systems used today in hospitals are stationary, wired, and cumbersome systems, several clinical applications can benefit from intelligent wearable, wireless, convenient, and comfortable solutions that provide high signal quality. Transferring the technology used in hospitals to users' homes would allow a large scale deployment of such clinical applications, significantly improving the therapies and knowledge by increasing the amount of data that can be collected and analyzed by the experts. For this reason, some portable EEG acquisition and monitoring systems have been recently developed, mainly meant for raw data transmission through wireless communication stacks [128]. Notable examples of recent wireless EEG monitoring systems are offered by Quasar [129], IMEC [130], Emotiv [131], NeuroSky [132]. However, these systems are only meant for consumer applications featuring 2 to 8 electrodes; the only one featuring a reasonable number of electrodes suitable for medical applications is g.tech [133].

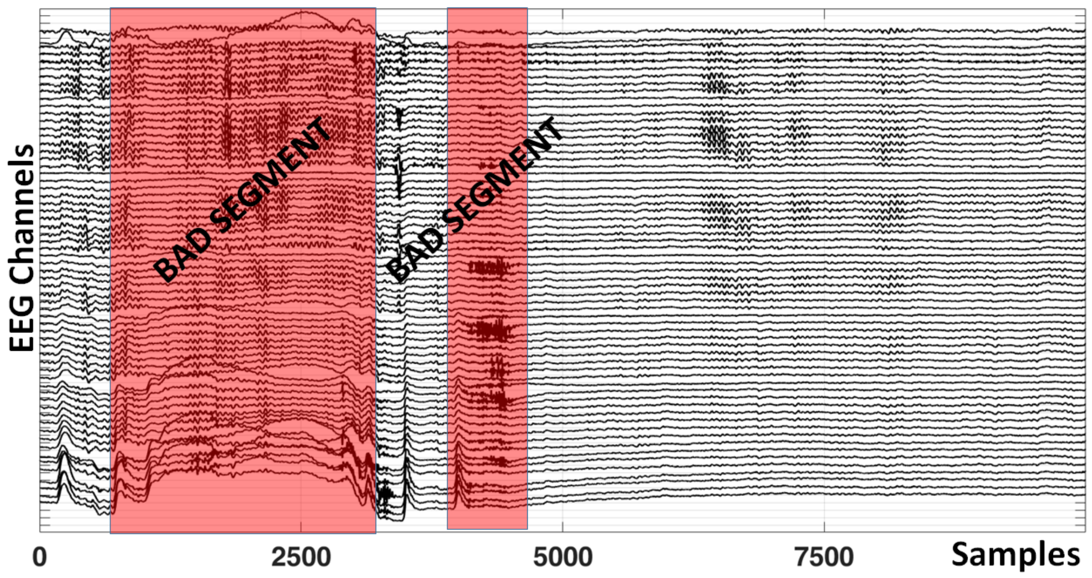


Figure 4.1: EEG traces with bad segments

In these systems tiny micro-controllers (MCUs) such as PIC-32 [134] or Cypress CY8C38 [128] are only used for transmission setup and control purposes. Therefore, due to the limited computational capability of the processors, these devices are only capable of transmitting raw EEG data, leading to a huge amount of wireless bandwidth. While Bluetooth (BT) is widely used for data transfer, the sheer amount of raw data transmitted during an EEG recording makes this protocol quite power hungry, especially for systems with a large number of electrodes (e.g., 64). Alternative more energy efficient protocols can be considered, such as Bluetooth Low Energy (BLE), even if at the price of lower throughputs and higher latency, therefore not always adequate for physiological data streaming [135] [136]. However, the use of a powerful and energy efficient processor into the system enables processing of data before transmission and consequent bandwidth reduction. In fact, only relevant content is sent with lower throughput, allowing the use of low-power protocols.

The lesson learned from the analysis of the literature is that the design of an energy efficient embedded system for real time EEG processing requires a multilevel design. Combining the algorithm and technology approaches, we designed an algorithm tailored for parallel, ultra-low power processing platforms [137][138]. Low-voltage operation and parallel computing is exploited to provide more than one order of magnitude of better performance and energy efficiency with respect to traditional MCUs. By exploiting close-to-sensor energy efficient data processing, it is possible to significantly reduce wireless transmission, designing more compact EEG monitoring systems and eliminating the need of an external host (e.g. a PC or a notebook), which can be replaced with a portable device such as a smartphone or a tablet with data visualization or device diagnostics.

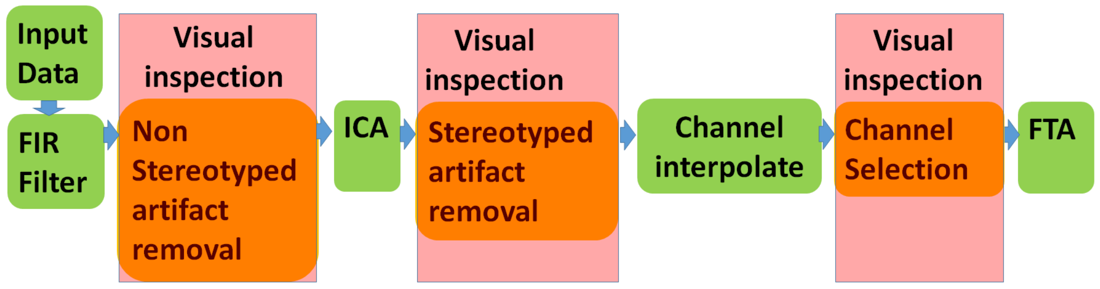


Figure 4.2: Block diagram of the standard analysis steps

4.2.2 Stimuli and EEG Acquisition Setup

Four right-handed (Edinburgh Inventory) native Italian speakers participated in the experiment (2 females; 21-29 years, mean age 23.3 years). All participants had normal or corrected-to-normal visual acuity, and reported no history of neurological or psychiatric disorders.

Stimuli consisted of a black and white 10x10 square checkerboard subtending approximately 15° by 15° of visual angle, on a uniform grey background, presented at a distance of 80 cm from the subject's eyes. Stimuli were presented with a sinusoidal on-off 100% contrast temporal modulation (black/white squares starting gray as the background, turning black/white at half cycle and ending gray at the end of the cycle) at 4 frequency rates (0.667 Hz, 0.8 Hz, 4 Hz, 12.5 Hz) in blocks of 28 cycles (42s), 32 cycles (40s), 160 cycles (40s), 360 cycles (28.8s), respectively, using the Psychtoolbox 3.0.12 for Windows in Matlab 8.0 (MathWorks Inc.). A sinusoidal contrast modulation was used because it generates fewer harmonics (i.e., responses at exact multiple of the stimulation frequency, reflecting the non linearity of the brain response [48, 139]) and because, since the on-off dynamics is smooth, it is a more pleasant and less fatiguing visual stimulation than a squarewave stimulation mode for the subjects. Subjects were asked to fixate at the center of a grey diagonal cross overlapped to the checkerboard. Each subject was presented with two series of blocks (presentation rates were randomized within each series).

The experiment was performed in an electrically shielded and sound-attenuated cabin. EEG was recorded with a BrainAmp amplifier (Brain Products, Munich) using 64 Ag/AgCl sintered ring electrodes mounted in an elastic cap (Easycap, Munich) and placed equidistantly according to the 10/20 system [140], with a vertex reference (Cz) and ground electrode in AFz. Electrode impedances were kept below 15kOhm. Data were sampled at 500Hz and analog filtered between 0.016 and 250Hz during recording.

4.2.3 Standard analysis

4.2.3.1 Pre-processing and artifact removal

In Fig. 4.1 a typical EEG trace is shown. Segments affected by artifacts are highlighted under red transparent boxes. Fig. 4.2 shows the block diagram of the standard analysis, based on visual inspection of the whole EEG trace. Continuous raw data were imported in the EEGLAB software [126] and band-pass filtered between 0.1 and 40Hz with the default EEGLAB filter (a Hamming windowed sinc FIR filter) to remove DC and high-frequency noise. Data were segmented in windows corresponding to the stimulation blocks. Each segment was visually inspected and portions containing non-stereotyped paroxysmal artifacts were discarded. Bad channels containing jumps larger than 200μV were discarded (no more than one per subject). To identify and remove stereotypical artifacts, the default EEGLAB ICA decomposition was computed on the concatenation of all segments. Blinks, eye-movements and other topographically localized artifacts were discarded by removing the corresponding independent components identified by ADJUST, an algorithm for automatic detection of artifacted ICA components [120]. Muscle artifacts were discarded by removing related ICA components identified by visual inspection of their topography and spectro-temporal profile. EEG signals in bad channels were interpolated with the EEG signals from neighbouring channels (standard spherical interpolation method in EEGLAB), and the resulting clean data were re-referenced to average reference. In summary, this standard procedure requires expert off-line intervention for the identification of bad data segments and artifacted ICA components, and computationally heavy ICA decomposition.

4.2.3.2 Frequency-Tagging Analysis (FTA)

To obtain a high frequency resolution with one bin centered on the stimulation frequency, for each tag frequency, EEG data from each channel were segmented into epochs of 18s (9000 time points, frequency resolution=0.0556Hz), 20s (10000 time points, frequency resolution=0.05Hz), 10s (5000 time points, frequency resolution=0.1Hz) and 10s (5000 time points, frequency resolution=0.1Hz) for the stimulation frequencies 0.667Hz, 0.8Hz, 4Hz, 12.5Hz, respectively, overlapping for half of their length. For each electrode, the Fourier transform $F_m(f)$ of each epoch was calculated using a Fast Fourier Transform (FFT) algorithm (MATLAB, Natick, MA) on m samples. The power spectrum was calculated from these Fourier coefficients as the average over epochs of the single-epoch power spectrum:

$$P_m(f) = F_m(f)F_m(f) \tag{4.1}$$

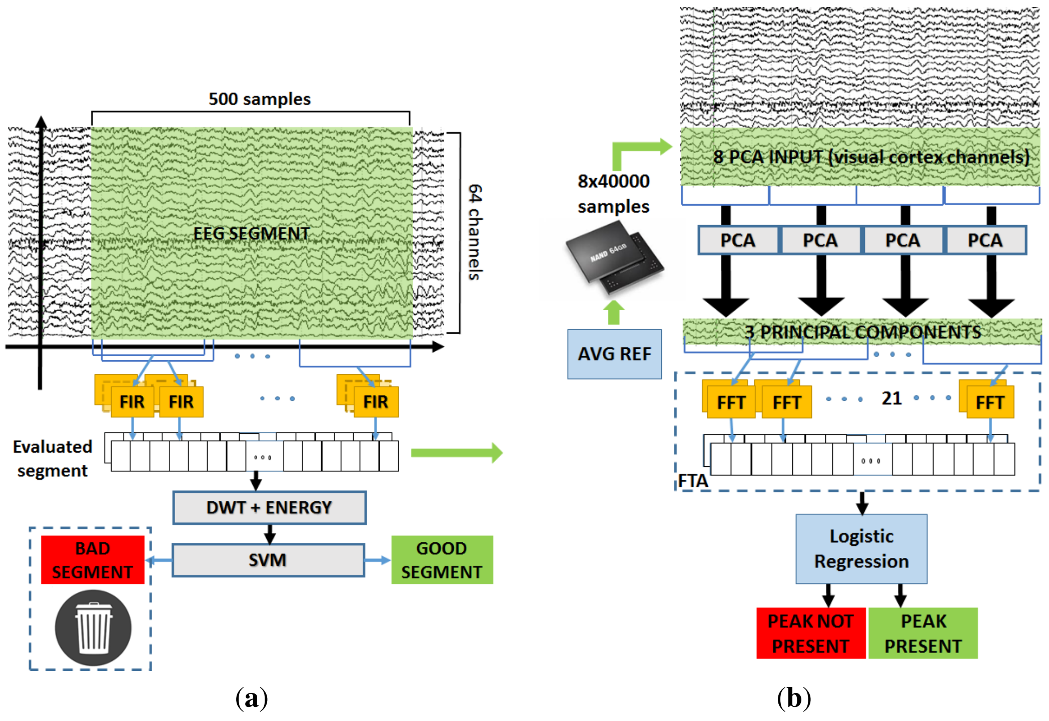


Figure 4.3: Block diagram of the automated artifact removal (**a**) and block diagram of the automated frequency detection (**b**).

Normalized power (NP) at the tagged frequencies (0.667Hz, 0.8Hz, 4Hz, 12.5Hz, respectively) was calculated as the ratio between the power spectrum at the tagged frequency and the average power spectrum at 2 neighboring frequency bins.

4.2.4 Automated Analysis

This sub-section describes the proposed machine-learning approach for automated analysis of EEG traces. The algorithm detects frequency peaks correlated to given visual stimula. The detection is composed of 2 parts described in the following: *Artifact Removal* and *Frequency Detection*.

4.2.4.1 Artifact Removal

In the *artifact removal* section a supervised classification algorithm detects and removes signal segments affected by artifacts that can degrade the analysis. A diagram of the algorithm is shown in Fig. 4.3(a). First, the signal is preprocessed using a bandpass FIR filter from 0.1 to 40Hz to remove higher frequencies. After filtering, the data is windowed and signal features are extracted calculating the DWT (Discrete Wavelet Transform) to provide information on the frequency content of the signal in the time domain.

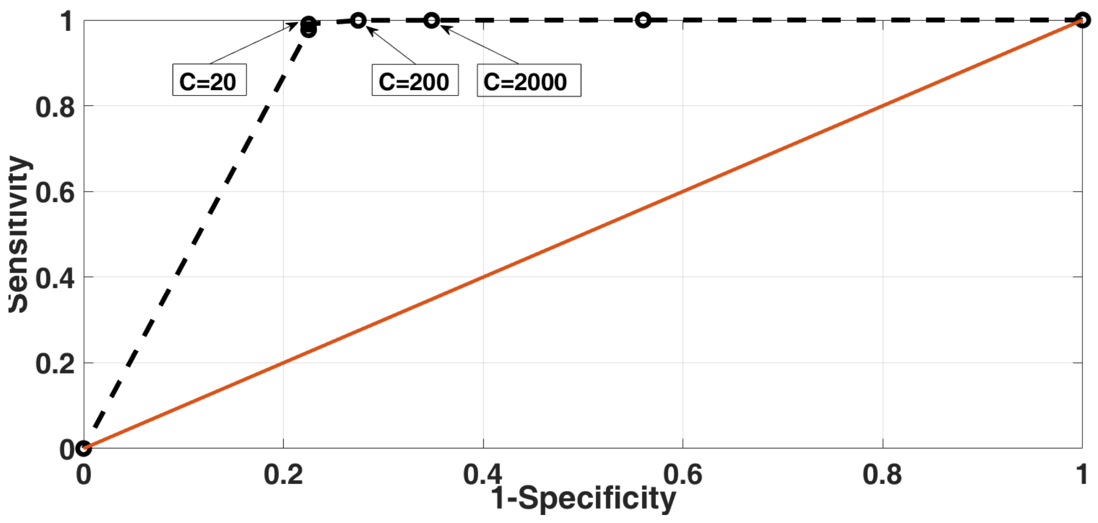


Figure 4.4: ROC curve for SVM performance evaluation.

Detail Coefficients of the DWT are used to extract the energy of the signal. Energy values are calculated at the 4 level of decomposition of the DWT according to (4.5) resulting in a vector of $1x256$ elements for each level.

$$\mathbf{E}_{D_{(n)}} = \sum_{i=0} \left|(D_{(n)}[i])\right|^2 \tag{4.2}$$

The energy coefficients are the input for a Support Vector Machine (SVM) classifier [98], widely used in biosignal embedded application by virtue of its theoretical robustness and implementation efficiency [16]. The classifier detects if the acquired segments are affected by movement or muscular artifacts and discards them. For the training phase 30% of samples belonging to the starting dataset and the related labels are used. Data is scrambled, downsampled and scaled to improve accuracy during prediction. The remaining samples of the dataset are tested during the classification phase. The SVM has 256 input features, thus we compared linear against Gaussian kernels to find the best trade-off between accuracy and computational cost. In this application, the accuracy of the two kernels is similar but the linear kernel needs a smaller computational effort, hence we selected it for our embedded implementation [141]. The tuning of the parameters of the classifier is performed calculating the ROC curve [99] obtained varying the parameter C of the classifier. For small values of C the separation hyperplane presents a smaller-margin while a larger-margin is obtained increasing C values. In Fig. 4.4 we present the ROC curve with a classification accuracy ranges between 85% to 96% and the best value shown is $C = 20$.

4.2.4.2 Frequency Tagging Analysis (FTA)

A diagram of the proposed algorithm is shown in Fig. 4.3(b).As in the standard analysis, data derived from the Artifact Removal stage are segmented into epochs of 18s (9000 time points, frequency resolution = 0.0556 Hz), 20 s (10000 time points, frequency resolution = 0.05 Hz), 10 s (5000 time points, frequency resolution = 0.1 Hz) and 10 s (5000 time points, frequency resolution = 0.1 Hz) for the stimulation frequencies 0.667 Hz, 0.8 Hz, 4 Hz, 12.5 Hz, respectively, overlapping for half of their length.

The target signals are generated by the visual cortex, located in the occipital region of the head. Hence, to extract the frequency information among the sensors placed in this region we apply the Principal Component Analysis (PCA), an orthogonal transformation that converts possibly correlated data distributed on p sensors into a set of linearly uncorrelated components ($l < p$). This algorithm is widely used in neural processing [142], to extract maximum variance components from a dataset or for dimensionality reduction [143] converting an input matrix into a new coordinates system through a linear transformation. In this application, we extract the maximum variance of the data from the 8 occipital EEG channels of the 10/20 system [140] maintaining 3 components with more than the 90% (to ensure this condition to be always satisfied, the number of PCs is fixed to 3) of the variance of the original input data.

This step of the processing chain has the double advantage to extract the frequency information among several sensors and to reduce the memory requirements of the computational framework, making it suitable for embedded implementation. The FT analysis is applied to the three components extracted by the PCA. This is a powerful method that allows to maximize variance in the first Principal Components through a linear transformation. Performing FTA on each channel shows that the target frequency is not always present or clear in all channels of the visual cortex and in this case they usually show small magnitude peaks. This means that clinicians should manually inspect the traces and select the best channels, i.e. the closest to the EEG response, to obtain a clear peak that allows to make an accurate diagnosis. PCA shows that the peak at the target frequency is always present and visible in the first PC and the magnitude is higher (up to 10 times in the 0.667Hz) compared to the peak obtained considering only data acquired from one channel. In this way, the computational effort dramatically decreases because FTA is performed only on three components rather than on the whole channels of the occipital region.

Since signals and FFT coefficients are affected by high variability caused by physiological or setup-dependent factors, a simple threshold detection does not guarantee adequate robustness for this application. Machine learning provides robust solutions such as logistic regression [144]; this algorithm is used to detect the presence of a frequency peak. The features used in this algorithm are the amplitude of the frequency peak, the normalized ratio between neighbours coefficients (i.e. the coefficients adjacent to the target frequency) and the ratio between the peak and the sum of the left and right neighbours, to evaluate if the peak is localized and narrowband.

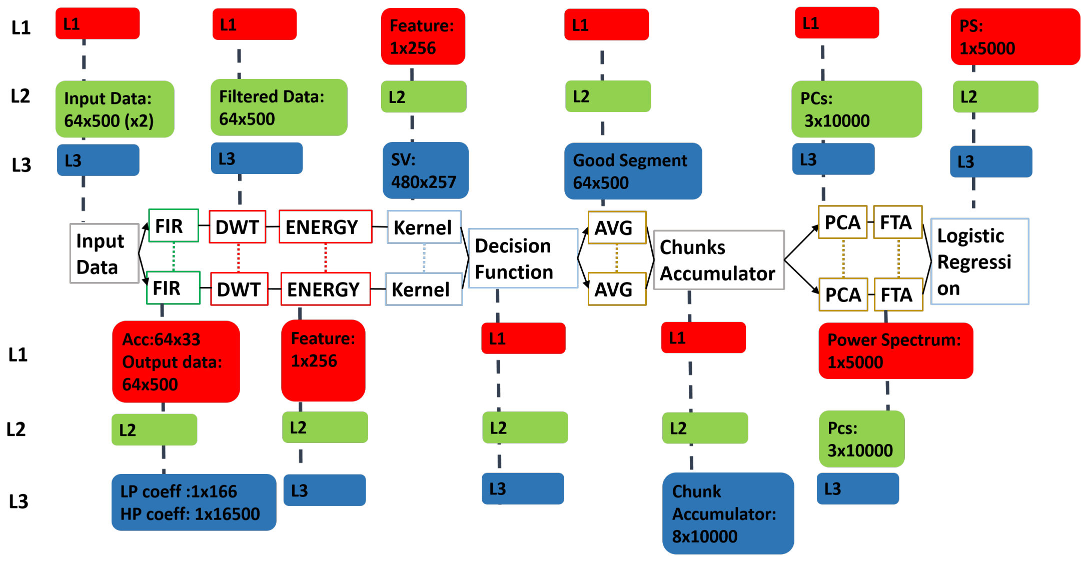


Figure 4.5: Automated Artifacts Removal and Frequency Detection computational Kernels.

Formulas of the 3 features are shown below, in (4.3)

$$\begin{aligned} x_0 &= p_n \\ x_1 &= \frac{p_n}{|p_{n-1} - p_{n+1}| + \varepsilon_d} \\ x_2 &= \frac{p_{n-1} + p_{n+1}}{p_n} \end{aligned} \tag{4.3}$$

where p_n is the target bin, p_{n-1} and p_{n+1} are the previous and the next bin respectively and ε_d is a given constant to prevent division by zero. Even in this case the classifier is trained off-line, once, using a training dataset. The weight parameters θ are obtained by application of the gradient descent technique, an iterative method to minimize convex functions. In this application, the peak recognition based on the LR reaches 92% accuracy. This accuracy values are aligned with SoA of machine learning applications for biomedical purposes [145–147].

4.2.5 Implementation on PULP

This section describes the implementation of the processing chain described in Section 4.2.4 on the PULPv3 architecture. The processing chain was decomposed in 6 sub-kernels, and each kernel was analyzed individually. The level of parallelism that can be achieved depends on the nature of the kernel itself. Fig. 4.5 shows a block diagram describing the processing chain emphasizing the two key aspects of the proposed implementation on a low-power parallel embedded system: parallelization scheme and memory requirements. In the following, we assume a configuration of the PULP platform with 8 cores and floating point units, unless differently specified.

Sampling data from 64 channels with a sample rate equal to 500Hz produces 64x500 samples per second, which have to be processed in real-time. We have assumed the data to be streamed into the SoC by 8 x 8-channels Texas Instrument ADS1298 ADCs through the SPI master port in 16-bit format, which are then stored in the L2 memory by the IO DMA. As first processing step, a low-pass FIR filter with 166 taps and a high-pass filter with 16500 taps are employed. Hence, before the processing can start, 1s of latency is needed to fill the samples buffer. After the first second of acquisition, a new 64x500 sample window is available every second. Hence, given that the order of the FIR filter is 16500 a delay of 33 seconds is necessary before the first meaningful filtered sample is produced. This leads an L2 memory requirement of 2 x 125kB to store the current and previous input samples windows, 66kB of L2 memory to store the FIR coefficients (i.e. *High-pass FIR coeff* and *Low-pass FIR coeff*), and 8kB to store the partial accumulations (i.e. *Accumulator*), which can be kept into the L1 memory. Every second, a new set of partial accumulations are computed by the processor, taking as input the current 64x500 input sample window.

To overlap the computation and data transfer (L2 to L1) phases, a double-buffering mechanism is employed, with 3 buffers, each one of 16kB (one for the input data, one for the output data, and one for the current data). The FIR filter is parallelized at block level (i.e. every core operates on 8 channels); once a new block of samples is available, threads compute outputs using past values of the filter and the latest samples, and store the accumulated data into the L1 memory buffer. Since the parallel FIR implementation is based on weighted sums without any dependency and the number of channels is a multiple of the number of cores, the workload is always perfectly balanced among cores and there is no need to use synchronization barriers leading to an almost ideal speed-up with respect to the single core execution. A copy of the filtered channels is also stored into the L3 memory, to be eventually re-loaded by the processing chain for frequency detection, if marked as good segment by the artifact removal algorithm.

After filtering the first samples window, DWT and Energy kernels (i.e. feature extraction) can be executed. Both kernels can be efficiently parallelized on the architecture since each thread can operate independently on a separate channel. Moreover, being the number of channels a multiple of the number of cores, the workload is again perfectly balanced, scales perfectly upon 2, 4 or 8 cores showing nearly ideal speed-ups. The output of the DWT and the Energy kernels are stored in a vector of dimension 1x256, (i.e. *feature*, 1kB) that can be allocated into the L1 memory.

The next processing step is the SVM classification. The main issue with the SVM algorithm is the large amount of memory required to store the support vectors that implement the model (i.e. *SV*). Each SV is composed of 256 values plus a coefficient, thus the total size per support vector is 1kB. The chosen model includes 480 SVs, requiring a storage of 482kB. In the proposed implementation, SVs are permanently stored in the L3 memory. With an SPI, data is transferred from the external L3 to L2 memory and from L2 to L1 with the DMA exploiting a double buffering polity. The parallelization of this kernel is also highly efficient since each thread independently operates on one of the 480 support vectors of the model, and the parallelism is so high (i.e. 480) that workload unbalancing does not impact performance.

Each time a new window of data marked as good segments (i.e. not discarded) is available, the filtered samples are loaded from L3 memory, and the mean value between all the channels is computed and subtracted from all the samples of each channel. Since the analysis is focused on the channels placed on the occipital area of the scalp, useless data can be discarded loading only data derived from the 8 channels of interest. This step is implemented by the Average Reference kernel, where each processor operates independently on the 64 samples of each time-step. Also in this case the computation on each core is completely independent, hence showing nearly ideal speed-ups with respect to the sequential execution. After Average Reference calculation, data is accumulated in the *Chunks Accumulator* (for a total of 320kB) and stored into the L2 memory.

In this application, PCA presents the most complex parallelization scheme, which is described in details in [143]. PCA is based on the Singular Vector Decomposition (SVD), to calculate eigenvalues and eigenvectors of a matrix. Both Householder's and Given's matrices are used to obtain the eigenvectors via bidiagonal transformation. The former part of the algorithm cannot be parallelized at block level, because there are dependencies on the previous iterations of the bidiagonal reduction, while the latter part reaches near-ideal speedups in the parallel execution. The matrix where PCA is performed has dimension 8x10000; the SVD part is performed on the covariance matrix of dimension 8x8, while Mean Value and Convariance Matrix and Principal Components are performed on a matrix respectively of dimension 8x10000 and 3x10000. The execution is dominated by this last two kernels achieving a total speed-up near to the ideal one. Considering the memory management, a single chunk of dimension 8x10000 is transferred to L2 memory through SPI. Then, with double buffering, smaller chunks are passed to L1 to continue the process. After PCA, 3x10000 samples are held in L2 memory (i.e. *3 Principal Components*). Then, FTA can be computed on each PC.

In the FTA stage, the most compute-intensive kernel is the FFT. The algorithm follows a butterfly diagram approach. Thus, in the parallel version butterflies are distributed among the cores and computed separately. Several synchronizations barriers are necessary at every step, so the speed-up decreases as the number of cores increases. For this reason, this kernel shows the lowest speed-up with 8 cores, but it does not affect the overall performance since FFT represent less than 1% of the entire processing chain. FFT is computed on overlapped chunks for each PCs. This data is stored in L2 and with double buffering are transferred to L1. After computing FFT, the *Power Spectrums* are summed in a vector allocated in L1 while the other data can be discarded.

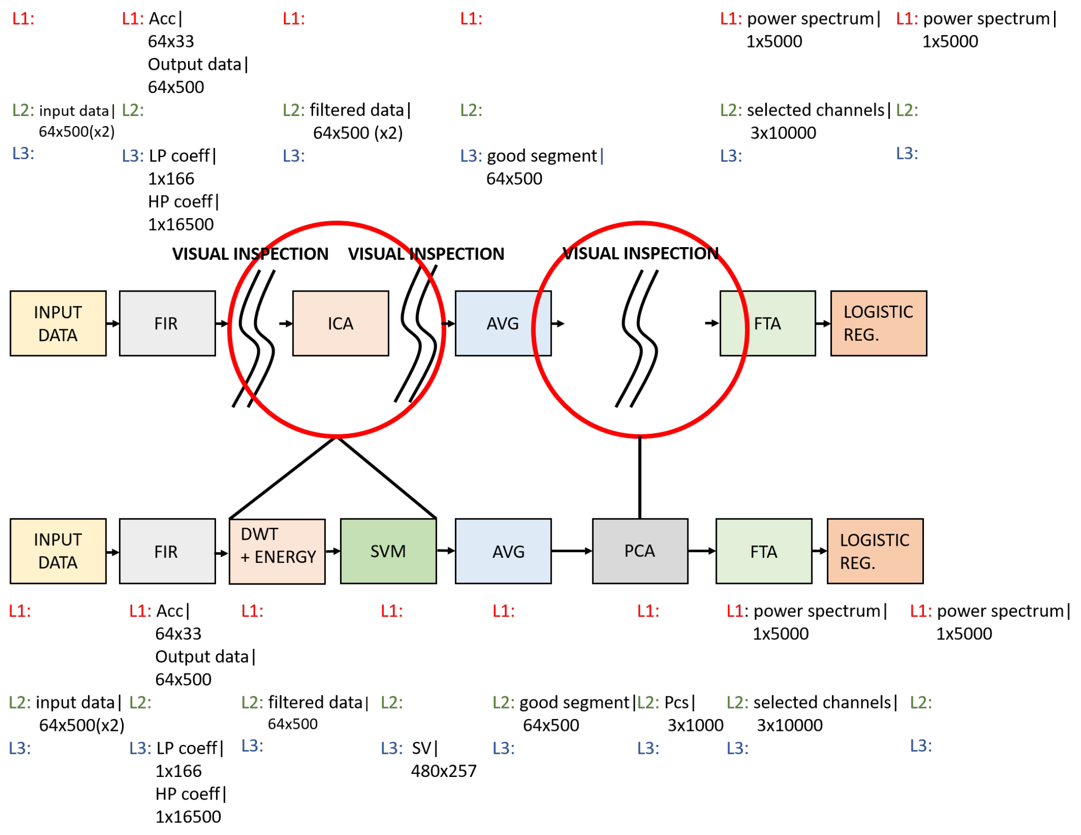

L1:
L1: Acc| 64x33 Output data| 64x500
L1:
L1:
L1: power spectrum| 1x5000
L1: power spectrum| 1x5000
L2: input data| 64x500(x2)
L2:
L2: filtered data| 64x500 (x2)
L2:
L2: selected channels| 3x10000
L2:
L3:
L3: LP coeff| 1x166 HP coeff| 1x16500
L3:
L3: good segment| 64x500
L3:
L3:
VISUAL INSPECTION
VISUAL INSPECTION
VISUAL INSPECTION
INPUT DATA
FIR
ICA
AVG
FTA
LOGISTIC REG.
INPUT DATA
FIR
DWT + ENERGY
SVM
AVG
PCA
FTA
LOGISTIC REG.
L1:
L1: Acc| 64x33 Output data| 64x500
L1:
L1:
L1:
L1:
L1: power spectrum| 1x5000
L1: power spectrum| 1x5000
L2: input data| 64x500(x2)
L2:
L2: filtered data| 64x500
L2:
L2: good segment| 64x500
L2: Pcs| 3x1000
L2: selected channels| 3x10000
L2:
L3:
L3: LP coeff| 1x166 HP coeff| 1x16500
L3:
L3: SV| 480x257
L3:
L3:
L3:
L3:

Figure 4.6: Comparison of the processing chain of the Standard and Automated Analysis with the most relevant structures for an estimation of memory requirements.

4.2.6 Experimental Results

4.2.6.1 Execution Time Estimation

To provide an estimation of the computational effort required for the application, a comparison between the execution time in both the standard and the automatic approach is performed. Fig. 4.7 shows the execution time of the two algorithms. As already mentioned, ICA decomposition is computationally intensive, hence it was excluded from the processing chain developed on the PULP platform. This choice was dictated by the necessity of finding a trade-off between accuracy and computational effort; this is mandatory during the development of embedded application, taking into account that Artifact Removal stage is enough accurate even without ICA.

The comparison is done considering the time required to execute kernels on Matlab, in seconds. As shown in Fig. 4.7, Automatic Approach results 6x faster than the Standard Approach. Furthermore, as shown in Fig. 4.6, in Standard Approach we are not considering the time needed to scroll and discard bad segments from the specialist, while in the Automatic Approach this task is performed on-line during the acquisition chain.

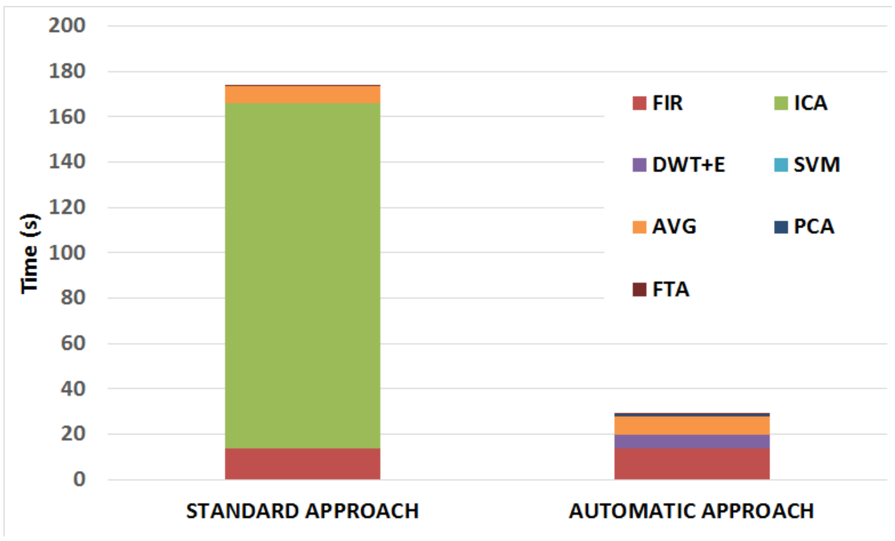


Figure 4.7: Estimation of execution times between Standard and Automatic Approaches.

4.2.6.2 Frequency Detection

Results of the FTA are presented in Fig. 4.8. The power spectrum of the raw data, without sample removal, is shown in red dotted lines with diamond markers, while the power spectra obtained with the standard analysis (Section 4.2.3) and automated analysis (Section 4.2.4) are shown in blue dashed (x markers) and black solid (round markers), respectively. For convenience we present, for each subject, the data of one low stimulation frequency (0.8Hz, left-hand column) and one high stimulation frequency (12.5Hz, right-hand column). While for the high stimulation frequency the resonant peak is evident even in the power spectrum of raw data, for the low stimulation frequency, the amplitude of the peak in the raw data is comparable to that of ongoing brain fluctuations or artifacts, therefore its identification is problematic (note in particular the raw data power spectra of subjects 2 and 4). The denoising effect of the standard analysis consists in removing a large part of unrelated fluctuations and focusing on channels containing the stronger frequency-tagged response. Therefore, even if the peak amplitude decreases, the power spectrum at neighbouring frequency bins decreases much more and flattens across the whole frequency range, resulting in a much higher normalized frequency-tagged response. The denoising effect of the automated analysis is almost as successful: the stimulation peaks are clear and detectable, even for the low stimulation frequency. An overview of the power spectra for all the stimulation frequencies averaged across the 4 subjects considered in the analysis (Fig. 4.9) confirms the practical feasibility of automated analysis: even though the power spectrum of the stimulus-unrelated activity is generally higher than the one obtained with the standard analysis, peaks at the stimulation frequency are also higher, resulting in a comparable peak detection efficacy, even for the low stimulation frequencies. The purpose of Frequency Detection is to understand if the brain replies to the visual stimulus properly, analyzing data acquired from the visual cortex of the subject. This means that the final response of LR helps technicians or

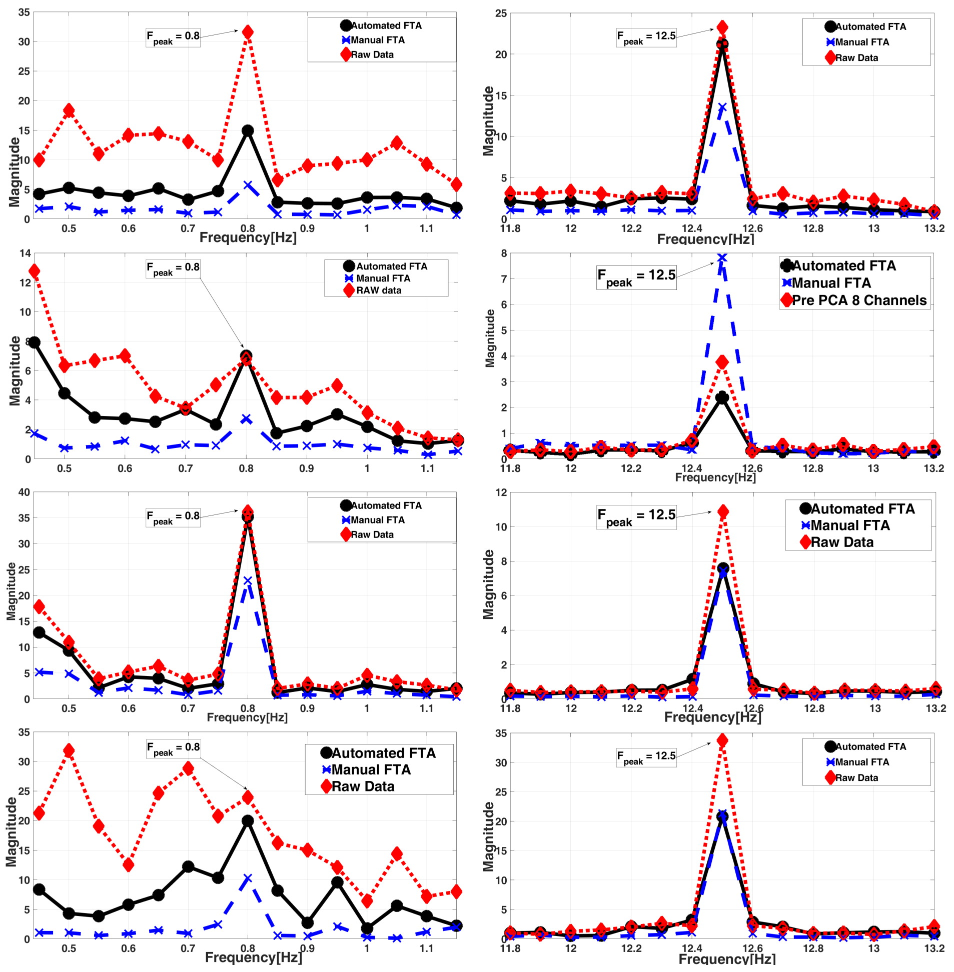


Figure 4.8: Peaks obtained from the visual stimulus at 0.8Hz and 12.5Hz. Graphs show values obtained from 4 subjects exposed to stimulus at the target frequencies.

medical staff in diagnosis of possible dysfunctions of the neurological activity (i.e. the peak is not localized or not narrowband). Thus, it can be considered as a support tool for clinicians to decide if an accurate analysis is needed to make a complete diagnosis.

4.2.6.3 Evaluation of Performance and Energy Metrics

Scaling the processing chain on a cluster of multiple cores allows to decrease the operating frequency required to achieve the real-time constrains, reducing significantly voltage supply. Thus, the quadratic dependency between supply voltage and dynamic power brings an important improvement in power density. This emphasizes a trade-off between the parallelization efficiency and voltage scaling. Execution time of the processing chain on the reference platform are shown in Table 3. Among all kernels that compose the application, FIR filter is the one that requires the majority of the computational effort. In fact, it represents the 97,88% of the execution time on PULP. As already said in the previous paragraphs, FTA kernel shows a lower speed-up compared to the others. The butterfly diagram approach used to compute FFT leads

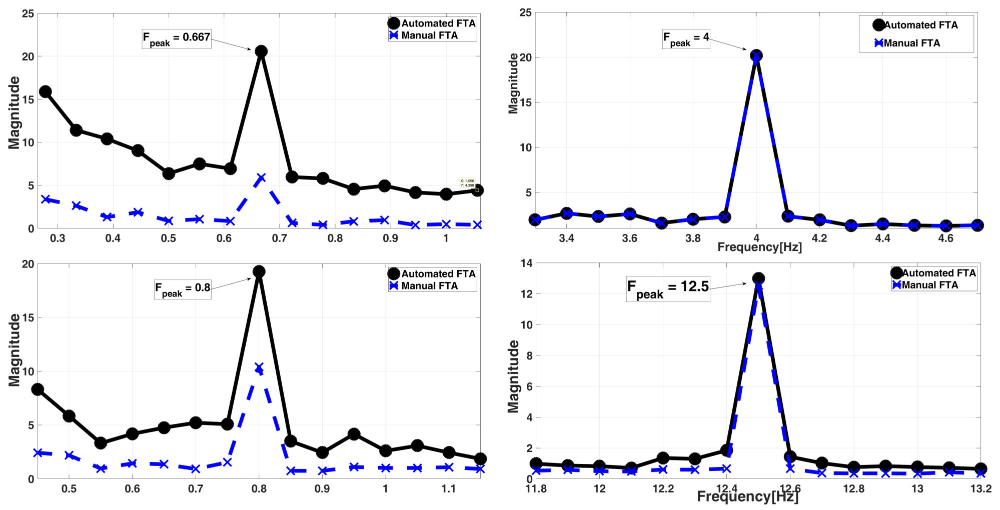


Figure 4.9: Peaks obtained from the visual stimulus at 0.667Hz, 0.8Hz, 4Hz and 12.5Hz. Graphs show mean values obtained from 4 subjects exposed to stimulus at the target frequencies.

	PULP 1 core		PULP 2 cores		PULP 4 cores		PULP 8 cores	
Kernel	MCycles	load %	MCycles	Speedup[a]	MCycles	Speedup[a]	MCycles	Speedup[a]
FIR FILTER	3303,67	97,88	1651,88	1,99	826,99	3,99	413,22	7,99
DWT+ENERGY	2,50	0,07	1,25	1,99	0,63	3,98	0,32	7,83
SVM	2,57	0,07	1,29	1.99	0,65	3,94	0,33	7,69
AVG REF	0,93	0,03	0,47	1,99	0,24	3,96	0,12	7,58
PCA	32,86	0,97	16,76	1,96	8,49	3,87	4,30	7,63
FTA	38,86	0,98	21,45	1,81	13,43	2,89	9,53	4,07
TOT	3380,86	100	1693,10	1.99	850,43	3.97	427,82	7.90

[a] Speed-Up with respect to single-core PULP paltform.

Table 4.3: Execution of automated frequency tagging analysis on embedded computing platform

to a parallelization scheme where a high number of synchronizations barriers among cores are necessary. The weight of this barriers grows as the number of cores increases. The load of this kernel represents the 0.98% of the overall processing chain, therefore the impact on the total performance is negligible. Due to the high number of cycles required to execute the process chain, a single core is not able to achieve the necessary operating frequency (i.e. 3.3GHz). The same constrains are worth for 2 and 4 cores (i.e. 1.6GHz and 830MHz). For these reason, a cluster of 8 cores is employed. With 8 cores an overall speed-up of 7.9x is reached, therefore, the application demonstrates to scale easily on higher number of cores.

Based on these information, PULP platform can fulfill computational effort and the real-time constrains with 8 cores with an operating frequency equal to 430MHz, at the supply voltage of 0,85V consuming an average power of 55mW. As an FRAM is used as external L3 memory, an estimation of load/store operations is necessary to understand the impact on power consumption. Considering that for an access in L3 0,162nJ are required, the average power consumed is

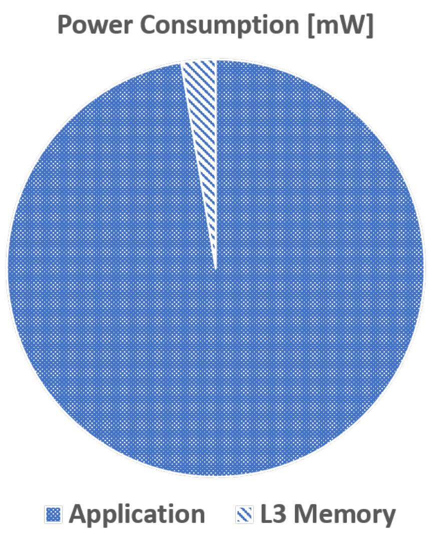


Figure 4.10: Estimation of the power consumption of the 8-core execution - application and L3 memory.

around 1,5mW. Thus, the total power consumption for the entire process chain is 56,5mW. Fig. 4.10 shows the contribution of the power required for the processing of the application and the power derived by the use of the L3 memory. The L3 memory is a non-volatile memory used to store data that need retention, and the application uses data stored in the L1 and L2 during the execution, without using the L3 memory intensively. Therefore, taking into account the power consumption and the supply voltage, approximately 66mA are required to correctly operate. Hence, assuming that this wearable device is provided with a common battery with a capacity of 2200mAh, a battery life around 24 hours can be guaranteed, with or without L3 memory power contributions.

4.2.7 Discussion

This chapter presented a machine learning-based, automated, embedded, and real-time EEG monitoring system targeting the analysis of the frequency tagging response in a wide range of frequency stimulations including both the standard range of SSVEPs (> 6 Hz) and the low-frequency range related to slower, higher-order neural processing (0.5-6 Hz). The proposed approach was compared with a standard method based on a mix of traditional signal processing chain (i.e. available in extraction and interpretation toolboxes such as EEGLAB) and the manual inspection of the EEG traces performed by an expert neuroscientist, on the same EEG signals from 4 subjects. It has been shown that by replacing the most compute-intensive parts of the traditional processing chain with a machine learning approach it is still possible to detect the presence of the same frequency peaks.

Although the standard analysis performs better in terms of signal cleaning, it is computationally intensive (i.e. because of ICA algorithm), partially manual (i.e. data scrolling and bad segments rejection) and therefore not adequate for a real-time embedded platform. Machine learning techniques are key elements for the system because they play an important role in Artifact Removal and Frequency Detection. The proposed solution achieves 92% correct peak detection; as already shown in Section 3.3.2 this result is in line with other medical studies present in literature.

Furthermore, its implementation has been evaluated on a parallel ultra low-power platform. The overall power budget obtained is under 56mW with 8-cores, thus making the system eligible for the future wearable deployment. The next challenges to be addressed include the generalization of the machine learning models, creating a model that can fit with different subjects without incurring in loss of accuracy. An other important challenge is to identify and remove single EEG channels that are recording bad segments of data (i.e. if the electrode is broken or not perfectly stick on the scalp) without discarding good data acquired by the other channels. All these challenges will converge in reducing the complexity of the processing chain for embedded implementation while improving the accuracy of the frequency detection. This paradigm is also suitable for a wide range of low-obtrusiveness applications in healthcare and rehabilitation scenarios.

4.3 Seizure Detection Algorithms for an Efficient Implementation on Embedded Devices

4.3.1 Related Work

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 [60], 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.

Table 4.4: Comparison with the state of the art of seizure detection implementation.

	Yoo J. [64]	**Chen W.M.** [148]	**Patel K.** [149]	**Verma N.** [150]	**Lee K.H.** [65]	**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	✓	✓

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 [61], 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 [62] 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. The majority 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 [64] 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 [151]. Another similar work is shown in [148]: 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 [149], a real-time seizure detection for ambulatory EEG signals is presented. In this implementation, 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 [150], 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 [65], 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 [152], where a 16-bit microcontroller [153] with an RISC architecture based on the standard MSP430 ISA (Instruction Set Architecture) is coupled with a coordinate rotation digital computer (CORDIC) [154, 155] 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

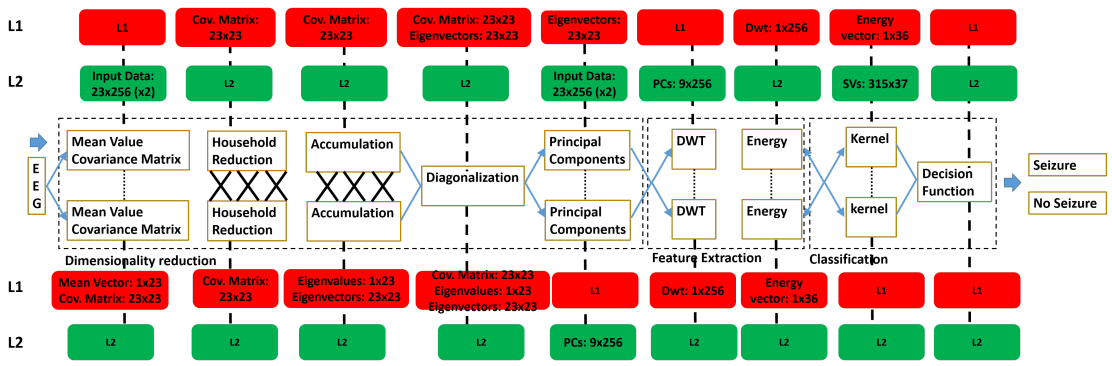


Figure 4.11: Seizure detection algorithm processing chain and memory requirement for each block.

scalability; and they are not fully reusable in other application scenarios. In Table 4.4, 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 [150] 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 [65], an inflexible SVM hardware accelerator has been employed to allow real-time operation within the power budget.

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 [60–62]. 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 [156]. In this chapter, an optimized implementation of a seizure-detection processing chain on PULP is presented, consisting of a dimensionality reduction (PCA and CS), feature extraction and classification steps [61, 157].

4.3.2 Principal Component Analysis-based Seizure Detection

Principal Component Analysis (PCA) is a commonly-used algorithm in the processing of EEG signals. 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:

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{P} \tag{4.4}$$

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 [158]. 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).

$$\mathbf{E}_{d_{(n)}} = \sum_{i=0} \left| d_{(n)}[i]) \right|^2 \tag{4.5}$$

The energy coefficients are used as input for the pattern recognition stage.

Among the different possibility, for this application we trained and validated a SVM. 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. In Figure 4.12, 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%.

4.3.2.1 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

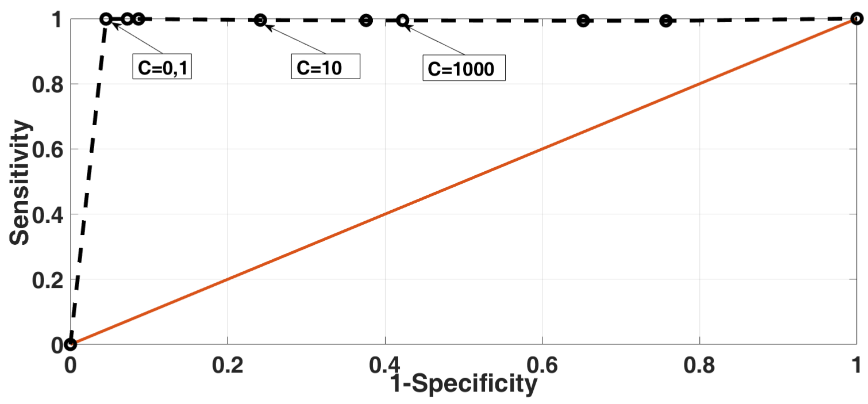


Figure 4.12: *ROC curve* for SVM performance evaluation.

8 cores. In Figure 4.11, a block diagram describes the processing chain with particular emphasis on parallelization schemes and memory requirements.

4.3.2.2 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 [159]. 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, as shown in *Tables* 4.6 and 4.7, 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 23×23 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.

4.3.2.3 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.

4.3.2.4 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.

4.3.3 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 [126]. 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 [160], 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 [161].

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:

$$((\mathbf{int})(\mathbf{x} * (\mathbf{int})(\mathbf{1} << \mathbf{FWL}))) \tag{4.6}$$

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

$$((\mathbf{float})\mathbf{y}/(\mathbf{float})(\mathbf{1} << \mathbf{FWL})) \tag{4.7}$$

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 [161]. 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:

$$((\mathbf{A} >> \mathbf{9}) * (\mathbf{B} >> \mathbf{9}) >> \mathbf{1}) \tag{4.8}$$

$$((\mathbf{A} >> \mathbf{7}) * (\mathbf{B} >> \mathbf{7}) >> \mathbf{5}) \tag{4.9}$$

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 [162]. 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 [162]. 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.

4.3.4 Experimental Results

4.3.4.1 Experimental Setup

An on-line dataset called CHB-MIT is used to take EEG data used in the simulation [163]. 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,

	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.

Table 4.5: Accuracy and Precision obtained from MATLAB and PULP platform floating- and fixed-point executions.

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 2.2b, 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.

4.3.4.2 Seizure Detection Accuracy

Table 4.5 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 + ENERGY kernel, 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 [164–166].

	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 + Cov.	1018	53.50	514	1.98	0	284	3.59	0	146	6.95	0.60
Householder Red.	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
TOT	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 4.6: Execution of floating-point seizure detection on the embedded computing platform.

4.3.4.3 Evaluation of Execution Performance

Tables 4.6 and 4.7 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× in the floating-point version and 3.44 in the fixed-point version. Analyze Table 4.6 first, which shows the results of the floating-point implementation, it can be noticed 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 4.3.3) and the difficulty to converge in the diagonalize kernel due to loss of precision.

Despite 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 2.3.1.1. This concept is highlighted in both Tables 4.6 and 4.7, 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

	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 + Cov.	995	42.2	536	1.86	0	284	3.50	0.40	143	6.97	0.60
Householder Red.	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
TOT	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.

Table 4.7: Execution of fixed-point seizure detection on embedded computing platform.

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%.

4.3.4.4 Hybrid Implementation Performance

Table 4.8 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 the execution time (i.e., number of cycles) is considered 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 4.8, 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.

Considering 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 4.9, 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

	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 + Cov.	1018	**995**	0.98	**514**	536	0.98	284	284	1.01	146	**143**	0.99
Householder Red.	**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
TOT	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 4.8: Execution time on the PULP platform.

	PULP 1 Core		**PULP 2 Cores**		**PULP 4 Cores**		**PULP 8 Cores**	
Kernel	E.Fl. (J) [a]	E.Fx. (J) [b]	E.Fl. (J) [a]	E.Fx. (J) [b]	E.Fl. (J) [a]	E.Fx. (J) [b]	E.Fl. (J) [a]	E.Fx. (J) [b]
PCA								
Mean + Cov.	**3206**	4409	1495	**1468**	1100	**789**	886	**471**
Householder Red.	**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**
TOT	7576	–	3548	5201	2634	3295	2279	2351

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

Table 4.9: Energy per frame on the PULP platform in J with 5-ms real-time constraint.

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.

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 4.13, 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.

4.3.4.5 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

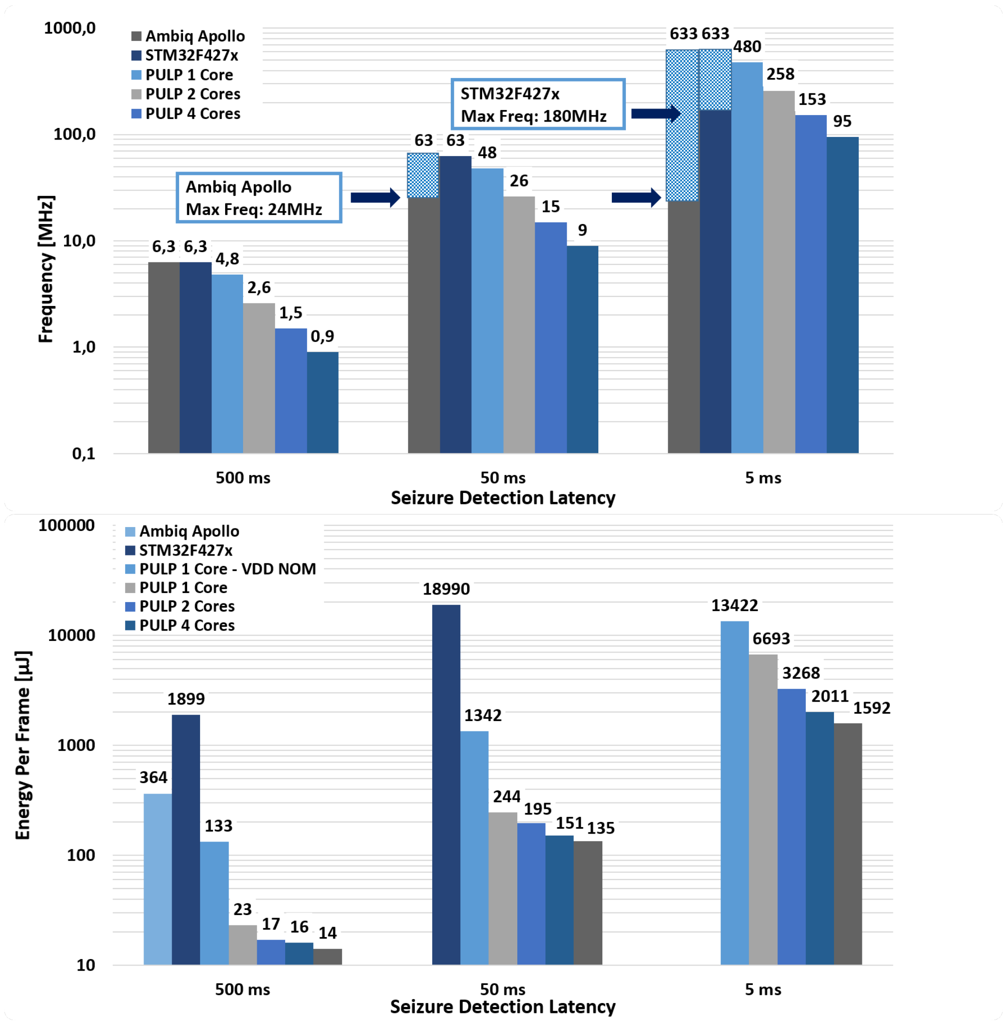


Figure 4.13: *Operating frequency* and energy per frame of the hybrid approach on the PULP and commercial MCUs.

processors are 115 W/MHz and 600 W/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 4.13 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 W, 274 W 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).

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 [167].

4.3.5 Discussion

This chapter 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, the optimized implementation of the seizure detection algorithm was presented, analyzing different solutions, which explore the performance/accuracy trade-off. The results of the 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 was 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

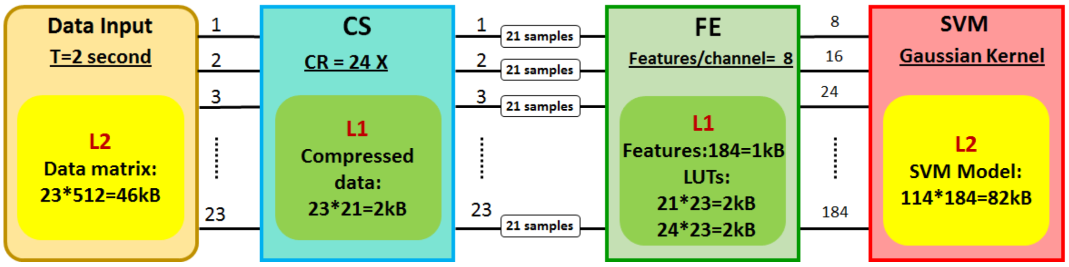


Figure 4.14: Seizure detection algorithm processing chain and memory requirement for each block.

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 has been 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).

4.3.6 Compressed Sensing-based Seizure Detection

During the last years, CS is gaining ground as a promising framework to tackle energy consumption and computational complexity issues in high dimensional and long-term monitoring of physiological signals. Nevertheless, CS-based acquisition techniques suffer from the reconstruction process, since it relies on the solution of an optimization problem that can reach $O(N^3)$ complexity. Extracting the desired information directly from compressively sensed signals avoids this problem and reduces the required energy budget of the application. In CS, signals are sub-sampled at a smaller sampling rate w.r.t. the Nyquist theorem. For this reason, a signal $\mathbf{X}_{N\times 1}$ is multiplied by a projection matrix $\mathbf{\Phi}_{M\times N}$, as shown in

$$\mathbf{Y} = \mathbf{\Phi} X \tag{4.10}$$

This results in a sub-sampled signal $\mathbf{Y}_{M\times 1}$, where $M \ll N$. To ensure a successful compression the signals should be sparse in some bases. Moreover, the projection matrix and these sparse basis have to be incoherent[168]. The amount of data reduction is quantified through the compression ratio (CR), which can be defined as

$$CR = \frac{N}{M} \tag{4.11}$$

where *N* indicates the dimension of the Nyquist sampled signal and M is the dimension of the compressed signals.

The feasibility of applying CS principles for EEG signals has been demonstrated in [169]. The EEG signals are sparse in several basis like Gabor, Discrete Wavelet and Discrete Cosine Transform (DCT). Random sensing matrices with a sufficient number of samples exhibit a low coherence with these basis. Pamula et al.[170] show that using a projection matrix containing only one nonzero entry per each row at random positions is a valid low complexity approach, that still provides good integrity of the compressed sparse signal. This matrix can be created by randomly choosing a subset of the rows of an identity matrix. The same projection matrix can be reused for compressing every epoch of the signal.

As shown in Figure 4.14 each channel of the EEG data is compressed over a window of 2 seconds with 50% overlapping between windows (1 second latency) with a Compression Ratio (CR) of 24x. In this implementation, the Least-squares Spectral Analysis (i.e. Lomb-Scargle periodogram) is used to extract the features of compressively sampled signals. Lomb-Scargle periodogram is a well-known procedure to generate a power spectrum and to detect the periodic component in random and unevenly sampled dataset. Given an *N* sample signal $\mathrm{X_k}$ sampled at $\mathrm{t_k}$ times, where k goes from 1 to *N*. The Lomb-Scargle periodogram (LSP) is defined in [171] and its final formula is shown in Eq. (4.12)

$$\begin{aligned} P_{LS}(f) = \frac{1}{2\delta^2} & \frac{[\sum_{k=1}^{N}(x_k - \tilde{x})\cos(2\pi f(t_k - \tau))]^2}{\sum_{k=1}^{N} cos^2(2\pi f(t_k - \tau))} \\ + & \frac{[\sum_{k=1}^{N}(x_k - \tilde{x})\sin(2\pi f(t_k - \tau))]^2}{\sum_{k=1}^{N} sin^2(2\pi f(t_k - \tau))} \end{aligned} \tag{4.12}$$

where $\tilde{x}$ and δ^2 are respectively the mean and the variance of the data and the τ is calculated from Eq. (4.13):

$$\tau = \frac{1}{4\pi f} \arctan\left(\frac{\sum_{k=1}^{N} \sin(2(2\pi f)t_k)}{\sum_{k=1}^{N} \cos(2(2\pi f)t_k)} \right) \tag{4.13}$$

The amount of LSP had been calculated for each epoch of compressed data in 8 frequency bands between 0.5 - 25 Hz.

The feature vectors obtained by the LSP are classified as seizure/non-seizure using the SVM, a supervised classifier which aims to find the optimal separation hyperplane between 2 classes. The separation hyperplanes represent the boundary between the 2 classes and it is defined by Support Vectors (SVs), weights (α_i), bias (b) parameters. Such values are used to classify a feature vector X according to the following equation:

$$C = \sum_i \alpha_i K(S_i, X) + b \tag{4.14}$$

	Accuracy(%)		Sensitivity(%)		Specificity(%)	
Subject	CS	PCA	CS	PCA	CS	PCA
1	91.4	92.9	96.0	98.0	91.4	92.9
2	96.0	94.2	93.5	95.5	95.9	94.2
3	94.5	90.5	94.5	92.6	94.3	90.4
4	96.5	97.4	96.1	100.0	96.5	97.4
5	98.4	99.0	100.0	94.9	98.4	99.0
Average	95.4	94.8	96.8	95.4	95.3	94.8

Table 4.10: Accuracy, sensitivity and specificity using data from 5 randomly chosen subjects of the CHB-MIT dataset.

	ARM Cortex M4 - work in [5]			ARM Cortex M4 - this work			1-core Mr.Wolf			8-cores Mr.Wolf		
Kernel	kCyc	ld%	$E[\mu J]^b$	kCyc	ld%	$E[\mu J]^b$	kCyc	ld%	$E[\mu J]^c$	kCyc	sp^a	$E[\mu J]^c$
PP	2600.00	82.00	1234.00	7.40	1.00	3.51	3.60	0.50	0.26	1.00	3.60	0.18
FE	192.00	6.00	91.10	195.60	27.30	92.83	140.00	20.00	10.29	19.20	7.29	3.40
C	369.00	12.00	175.10	512.00	71.60	243.00	553.00	79.40	40.66	83.90	6.59	14.87
TOT	3165.00	100.00	1502.21	715.00	100.00	339.34	696.60	100.00	51.17	104.10	6.69	18.45

[a]Speed-Up with respect to single-core Mr.Wolf paltform [b] 168MHz@1.8V [c] 110MHz@0.8V

Table 4.11: Proposed implementation on both ARM Cortex M4 [5] and Mr.Wolf. PP, FE, C and TOT stand for pre-processing, feature extraction, classification and total, while kCyc, ld, sp, E stand for thousands of cycles, load, speed-up and Energy.

where k denotes the kernel function. The model for the classification was computed off-line on MATLAB and the parameters of the training function were tuned through a k-fold cross-validation, leading to the choice of Gaussian kernel, c parameter between 10-100 and sigma between 10-20 (depending on the subject).

4.3.6.1 Experimental Results

The algorithm is evaluated on 5 of the 23 subjects included in the CHB-MIT [172] dataset. Data are obtained with a 23 electrodes setup on the scalps of epileptic pediatric subjects following the 10-20 System [140]. The EEG signals are sampled at 256 Hz, through a 16-bit ADC. The subjects used for the test were randomly chosen among the dataset. The processing chain was first tuned and executed on MATLAB, verifying the accuracy, sensitivity and specificity.

For the training of the classifier, the 25% of seizure epochs and the same length of non-seizure epochs are used. The remaining epochs are used for testing and validating the model. As shown in Table 4.10, the algorithm reaches on average 95.4% accuracy and 96.8% and 95.3% respectively for sensitivity and specificity, computed as

$$\begin{aligned} Sensitivity &= \frac{TP}{FN+TP} * 100 \\ Specificity &= \frac{TN}{FP+TN} * 100 \end{aligned} \tag{4.15}$$

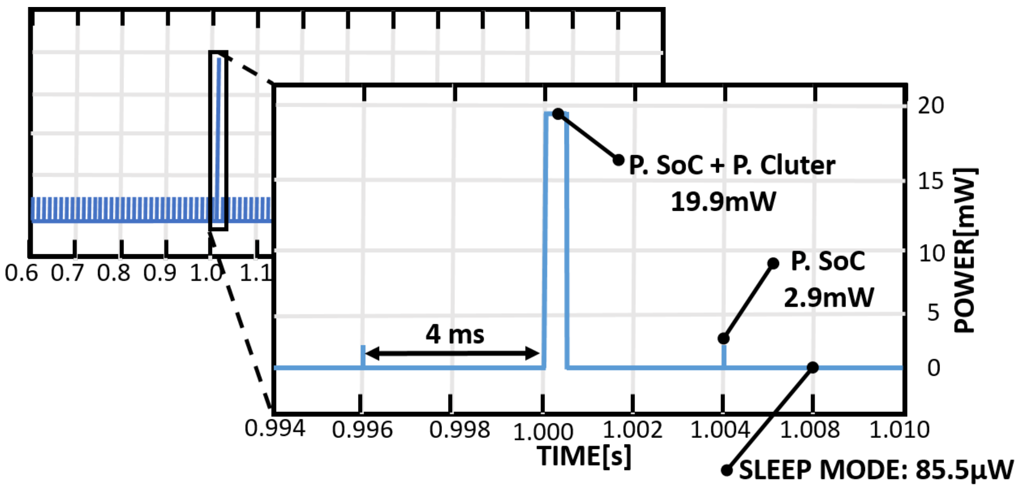


Figure 4.15: Mr. Wolf (8-cores) Power consumptions: idle (deep sleep), only the acquisition (SoC), acquisition and elaboration (SoC+Cluster).

where FN, FP, TP and TN stand for False Negative, False Positive, True Positive and True Negative. After the validation, the seizure detection algorithm was implemented and tested on Wolf and on a ARM Cortex M4-based commercial MCU.

4.3.6.2 Parallel Implementation on PULP

A block diagram of the seizure detection algorithm is shown in Figure 4.14, providing details of the whole processing chain and in Table 4.11 the execution time (clock cycles) and energy analysis of the seizure detector on the target platforms are summarized.

The CS kernel requires a small amount of clock cycles to be executed. In fact it represents only 1% of the overall execution time. The input data with the size of 46 kB is stored in L2 memory. The compressive sensing block randomly takes 21 samples out of the 512 input data for each channel (21x23=2 kB). From the Table 4.11, it is noticeable that, when executing the CS kernel over multiple cores, even if the workload is perfectly balanced among the cores, the speed-up tends to saturate. This is mainly due to the fact that random accesses to L2 memory (15 clock cycles for each access) are required. Thus, due to the randomness of these accesses, the double buffering would have no impact (or even negative) on the performance.

After CS, the Feature Extraction (FE) kernel is performed. This kernel constitutes around the 20% of the total load. The frequency power of compressed data are calculated using LSP function implemented for the frequency bands between 0.5-25 Hz. Moreover, since the frequency and the time stamps are constant in this application, the sine and cosine functions of LSP are implemented with Look Up Tables (LUT). The dimension of these LUTs are 24x23 and 21x23 (4 kB). The FE kernel demonstrates a strong predisposition to scale among all the cores, allowing to exploit parallelism. In fact, passing from the execution with 1-core to the 8-cores Wolf a 7.3x improvement is obtained.

The most time consuming kernel of the processing chain is the SVM on both ARM Cortex M4 and Mr.Wolf (71.6% and 79.4% respectively). The model of the SVM derived from the off-line training on MATLAB is stored in L2 memory (82 kB) and it is transfered from L2 to L1 memory via DMA with double buffering. SVM reaches 6.6x speedup with 8-core Mr.Wolf platform. The non-ideal speed-up of the SVM is caused by workload unbalance and by the final classification stage which is executed sequentially due to the lightweight workload (i.e. few hundreds of cycles), not justifying the overhead of an additional OpenMP parallel region.

4.3.6.3 Evaluation of the performance

Fig. 4.15 shows the power contributions during different operative states of the system. The total memory footprint for the implementation of the algorithm on Mr.Wolf platform is equal to 135 kB, hence we need 3 blocks of L2 memory to store all the data and the code. The amount of power used in sleep mode is 72 μW for the RTC and 3x4.5=13.5 μW for memory retention, which leads to a total power of 85.5 μW. The sampling frequency is equal to 256Hz, thus the system requires 2.9 mW (SoC Power) to acquire a new sample each 4ms. Furthermore, after the first second of execution, the cluster elaborates the acquired samples in 1.04 ms. At this stage, the total power is around 19.9 mW and it derives from the sum of both Soc and Cluster Power.

Table 4.11 shows the performance and the energy consumption measurements on the ARM Cortex M4-based (we targeted our implementation on a STM32F4-DISCOVERY board) and Mr.Wolf. In terms of execution time, we obtain a 4.4x improvement comparing the previous implementation with PCA and the one with CS. The ARM Cortex M4 operates at 168MHz at 1.85 V, thus in both the implementations the entire execution is performed with a latency less than 20ms. The most important aspect to consider is represented by 4.4x gain in the energy consumption obtained changing the pre-processing kernel (i.e. from PCA to CS). Moreover, passing from ARM Cortex M4 to 1-core Mr.Wolf give us a great improvement in term of energy consumption (6.6x at 110 MHz@0.8 V), even if the number of clock cycles are similar. This gap increases exploiting the parallel programming, leading to a 2.8x passing from 1 to 8-cores Mr.Wolf and 18.4x with respect to ARM Cortex M4.

Comparing our implementation with a similar work described in [66], which uses compressing sensing and linear features with different classifiers in multi-core platform (PENC), our algorithm shows better results in term of energy consumption and accuracy. In fact, they achieve a sensitivity of 81.8% and specificity of 93.9% for Nyquist-domain seizure data which degrade 2.07% and 2.97% for a CR=16x respectively. In addition, the energy consumption of their design when they use LR and SVM as classifiers are about 1.175 mJ and 0.0018 mJ with CR of 16x, respectively.

4.3.7 Discussion

This chapter presents a real-time, scalable and flexible seizure detection algorithm implemented on a Parallel Ultra-Low-Power (PULP) platform. This approach takes advantage of extracting the feature vector directly from compressively-sensed data to reduce the computational complexity. Extracting proper features in the compressed domain and using an SVM classifier lead to an average sensitivity of 96.8% and accuracy of 95.4%. Using CS instead of the PCA dimensionality reduction approach leads to an improvement of 4.4x in performance and energy consumption. Furthermore, implementing the algorithm on multi-core platform speed-ups the seizure detection algorithm by 6.7x with respect to sequential implementation. Moreover, the amount of energy consumption degrades 2.8x using the 8-cores Mr.Wolf. As the SVM kernel is the most time-consuming part of the processing chain (around the 80% of the overall load), implementing a configurable accelerator for this kernel can further reduce the system power consumption, and it can be the subject of a future work.

Chapter 5

ECG Applications

5.1 ECG Signals

5.1.1 Signal Description and Acquisition

The last class of biosignals explored in this thesis is the electrocardiogram (ECG). The ECG is the result of the recording of the electrical activity of the heart using electrodes placed on the skin of the subject. During the cardiac cycle (i.e. heartbeat), there is a depolarization phase followed by repolarization of the cardiac muscle that leads to a small electrical change. The orientation of the electrodes respect to the dipole ends leads to a different intensity of the detected voltage, while the amplitude is directly proportional to the mass of the tissues involved in creating the dipole. A collection of these electrical changes gives an idea about the status of the cardiac activity for the diagnostic of cardiac abnormalities or simply for the monitoring in particular conditions (i.e. during sports). The cardiac cycle starts with a fire that is too small and, thus, not detected by the electrodes. This is produced by the sinoatrial node in the right atrium. Then, there is a coordinate depolarization of the left and right atria that produces the P wave. The duration is between 80-100 *ms*, and after this time the signal returns to baseline. Roughly 160 *ms* after the P wave onset, the right and left ventricles begin to depolarize, resulting in around 80 *ms* of QRS complex. The end of the QRS complex corresponds to the end of the repolarization of the atria. This effect is not recorded by the electrodes, because it is covered by the ventricular depolarization. Now, the signal returns to the baseline until the repolarization of the ventricular, giving rise to the T wave [173].

The ECG signal is recorded computing the potential difference between two electrodes placed directly on the surface of the skin. The recorded waveform depends on the amount of cardiac tissue involved and the orientation of the electrodes with respect to the dipole in the heart. Typically, the ECG signal depends on the acquisition of a number of different electrode

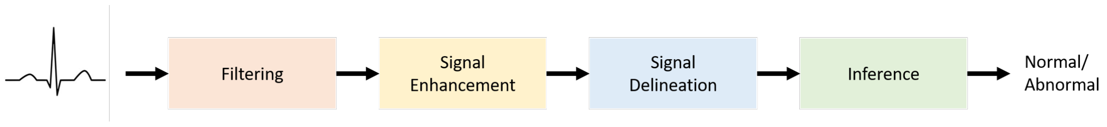


Figure 5.1: Modules for a typical ECG application.

positions or configurations and, for this reason, the process has been standardized. This signal consists of small amplitude voltages (±0.5 V) with high offsets (±300 V) and noise (50/60 Hz from the power lines).

5.2 Modular Design and Optimization of Kernels for the Development of ECG-based Applications

Population aging is causing a fast growth of the incidence of noncommunicable diseases (NCDs), for instance, cardiovascular diseases (CVDs), which are becoming the major cause of death (globally). The improvements in remote health monitoring and self-diagnoses, early intervention and prevention require the involvement of new technologies for the processing and the inference of features extracted from the ECG signals to make these devices effective. In particular, these devices should reduce considerably the intrusiveness in the users' daily life, being wearable and easy to use, ultra-low power and with contained costs.

For the intrinsic characteristics of the ECG signals, the processing usually does not require a very high computational complexity and, thus, the algorithms can be highly optimized for the single-core execution. With the new era of ultra-low power multi-core architectures for embedded devices, the performance can be further optimized splitting the workload among multiple cores, with different kind of parallelization depending on the characteristics of the targeted applications. Additionally, among all the possible software optimization, hardware accelerators can be specifically designed for the execution of specific tasks and integrated on the system to further improve performance, reducing the latency, and energy efficiency, increasing the battery lifetime.

Fig. 5.1 shows modules that can be assembled to compose a typical ECG-based biomedical application such as, for instance, Noncommunicable Diseases (NCDs) long-term monitoring. Typically, the first step of a ECG-based processing chain is the filtering to remove high frequency noise, baseline wandering and muscles noise. After filtering the signal, the second module usually includes processing to enhance specific features of the signal or to combine different leads into one in the case of multi-leads acquisition systems. Moreover, another common module is the signal delineation, which allows to extract information related to the main waves (i.e. QRS complex, P and T waves) from the ECG signal, important information used as features in the majority of the applications. Finally, the inference module makes use of features extracted from

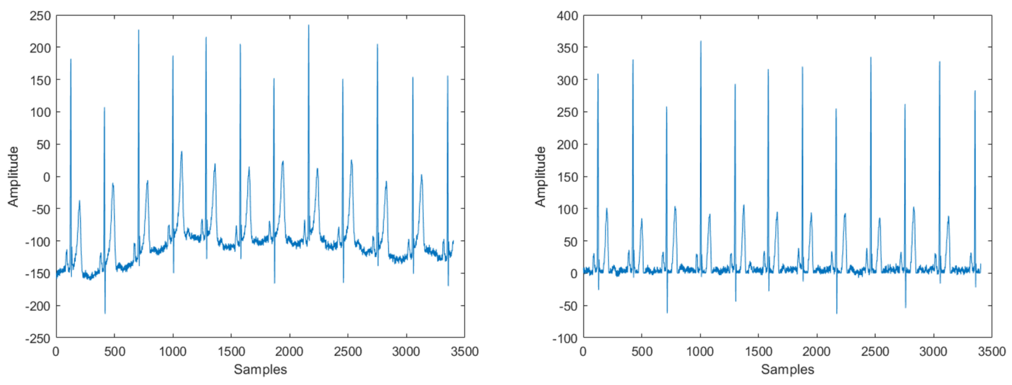


Figure 5.2: Left: Raw ECG signal. Right: Filtered signal using Morphological Filtering.

the ECG signals to predict an outcome (i.e. the occurrence of a pathology) using classifiers or regression techniques. Several techniques can be exploited to highly optimize these kernels on modern platforms, allowing to save energy and, thus, to the extension of the battery lifetime for wearable devices.

5.2.1 Filtering

The noise coming from different sources (i.e. baseline interference) dramatically affects the ECG signals. In literature, several techniques exists for the filtering of the ECG signal[174] and, among them, there is the morphological filtering (MF). This technique extracts the signal baseline based on the shape of the original signal to be removed. This method can be used to filter a single or multi-lead ECG. Through consecutive basic operations, named erosion and dilation, two other functions are created (i.e. opening and closing). The first function (opening) removes peaks from the ECG signal (removes P, R, and T), while the closing operation removes the Q and S. The result of these two operations is an estimation of the baseline drift that is subtracted from the original signal. Fig. 5.3 shows one ECG signal before (*left*) and after (*right*) applying the morphological filtering.

Typically, the ECG-based applications are based on signals acquired by multi-leads (e.g., 3 - 12 leads) to have more information about the status of the electrical activity. The execution time of this module is data dependent and full of condition and break statements. Thus, it does not demonstrate a high predisposition for data level parallelization. For this reason a multi-lead parallelization is the best approach to obtain and efficient parallel execution. The main issues related to this approach occurs when the number of leads is not a multiple of the number of cores in the cluster, leading to workload unbalanced.

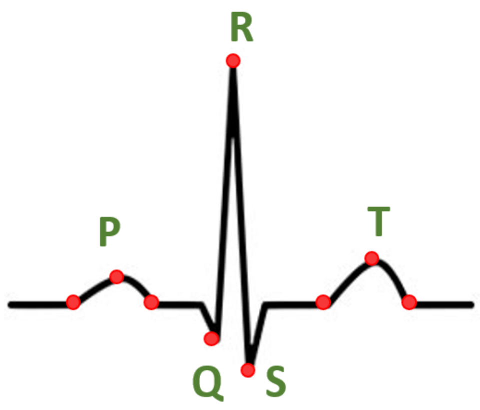


Figure 5.3: ECG-signal delineation. The figure highlights the QRS, P and T waves with the onset and offset of the waves.

5.2.2 Signal Enhancement

An important step for the processing of the ECG signals involves the signal enhancement and/or the leads combination. In literature, several techniques for the signals enhancement are described, and, among them, there is the Relative Energy (Rel-En) [175]. This kernel is used to amplify the QRS complex, extracting the energy of specific windows to heighten the R peaks, as the energy is larger when a R peak occurs. While, in the case on which multiple leads are involved in the processing, the leads combination combines all the leads into one to have a unique perspective of the electrical activity. In the case of applications that use data from a single ECG lead or for modules in the processing chain that are composed of almost pure sequential code, a window parallelization is an effective way to exploit parallelism. In the case of Rel-En, the window can be divided into smaller windows of data, and each core can start from the first sample of each sub-window to execute the sequential core in parallel to the others as explained in [176]. In this way, the computational workload is equally split among the cores, adding an overlap with respect to the single core version to ensure the correctness of the results.

Root Mean Square (RMS) is a signal enhancement technique applied on a buffer of data. In an ECG-based application, this is used to combine a multi-lead into a single-lead signal. This implementation computes the sum of squares of the samples of the different leads and then applies a square root to the result [177]. Since the elaboration of the samples of all the leads is independent, the workload is split equally among all the cores of the cluster using a data-level parallel paradigm.

5.2.3 Signal delineation

The ECG signal structure combines several waves including QRS complex, P and T waves. The QRS complex detection (or just the R peaks), is important for the implementation of different kind of applications because effective in many NCDs or other abnormal conditions. Hence, many approaches allow the detection of QRS independently, skipping the points related to the other waves. For instance, the REWARD [175] approach makes use of amplitude thresholds to isolate the R-peaks, requiring a low computational workload. Sometimes, the T wave of the ECG signals is predominant on the R peak. This approach defines other physiological parameters, avoiding miss-detections.

Other techniques delineates the rest of the points of the other waves. A low complexity method allows to find these points starting from the assumption that the main waves are positive. This assumption can be ensured by the RMS combination among multiple leads or through a leads selection phase [178]. With this technique, the Q and S points are easily found as the minimum within an interval near the R peak. While, the P and T peaks are the maximum within windows between two consecutive R peaks. Then, the onset and offset of the P and T waves are the minimum Euclidean distance between the original waves and their linear approximation. The point with the minimum Euclidean distance that intersects the isoelectric line is the onset/offset of the wave. For the delineation of the ECG signal, many applications require to execute the same operations for each beat. Hence, a possible approach to get a good level of parallelism, is to divide the workload inside the cluster based on this assumption. Thus, each core performs the same operations on a different beats (or sub-group of beats). Considering the approach described in [178], the signal is comprised between two R peaks. The workload varies from core to core due to the natural variability of the RR intervals (i.e., heart rate), leading to possible workload unbalanced that degrades the performance.

5.2.4 Inference

The last module that can be included in a typical ECG-based application is the inference. It can be used to automatically detect a medical condition from the ECG signals processing, important for the monitoring of the heart activity or for the detection of the presence of a pathology. For instance, there are several types of arrhythmia that change the electrical signal acquired from the abnormal heart activity. The early detection of these abnormalities in the heart beats helps during the treatment, preventing further complications. For instance, abnormal beats can be classified using random projections and a neuro-fuzzy classifier[177]. This kernel can be parallelized splitting the different heart beats among the cores, in the same way described for the ECG signal delineation (i.e. each core elaborates data related to different heart beats independently).

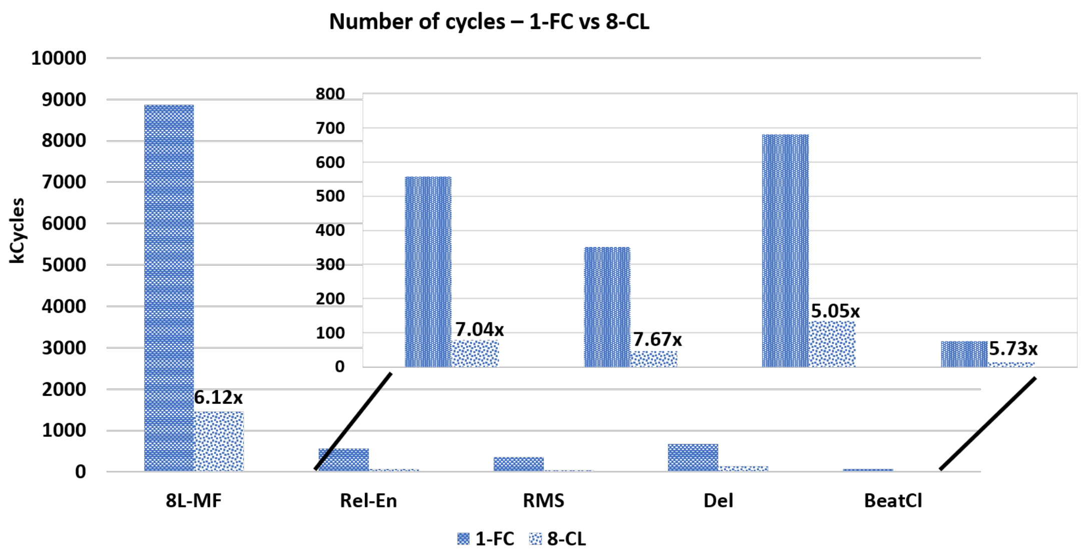


Figure 5.4: Execution time in the ECG modules.

5.2.5 Experimental setup and Results

In this section, we show some preliminary results obtained during a work in collaboration with the Embedded Systems Laboratory (ESL) from Ecole Polytechnique Fédérale de Lausanne (EPFL).

To test all these kernels, we design a test bench for each module with the input signals taken from specific on-line data sets used in the benchmarks. For the filtering, signal enhancement and signal delineation modules, we consider signals from the Physionet QT database (QTDB) [179]. We choose four signals from the QTDB, representing the worst, the best and two average cases in terms of combination of noise and shape of the three ECG waves. Finally, for the inference module, we consider the MIT-BIH Arrhythmia Database (MITDB) [180]. Even here, for the analysis we choose four signals as the worst, the best and two average cases in terms of percentage of abnormal beats over the total annotated beats.

Fig. 5.4 and 3.8 show respectively speed-ups obtained considering the 1-core FC and 8-cores CL execution on Mr. Wolf and the derived energy saving. The first kernel, the 8-leads morphological filtering (8L-MF) gives a speed-ups of 6.1× with an energy saving of 35%. The loss in speed-up derives from the need to allocate some variables and structures on the L2 memories for retention to avoid multiple execution of the initial filling of the buffers used by the filter and, thus, the derived transients. For this reason, when we perform the morphological filtering with the FC, the accesses to the L2 memory have no impact on the performance (1 cycle latency), while, when we execute the morphological filtering with the cluster, the latency for the accesses in L2 have a bigger impact (around 15 clock cycles) leading to a loss in speed-up.

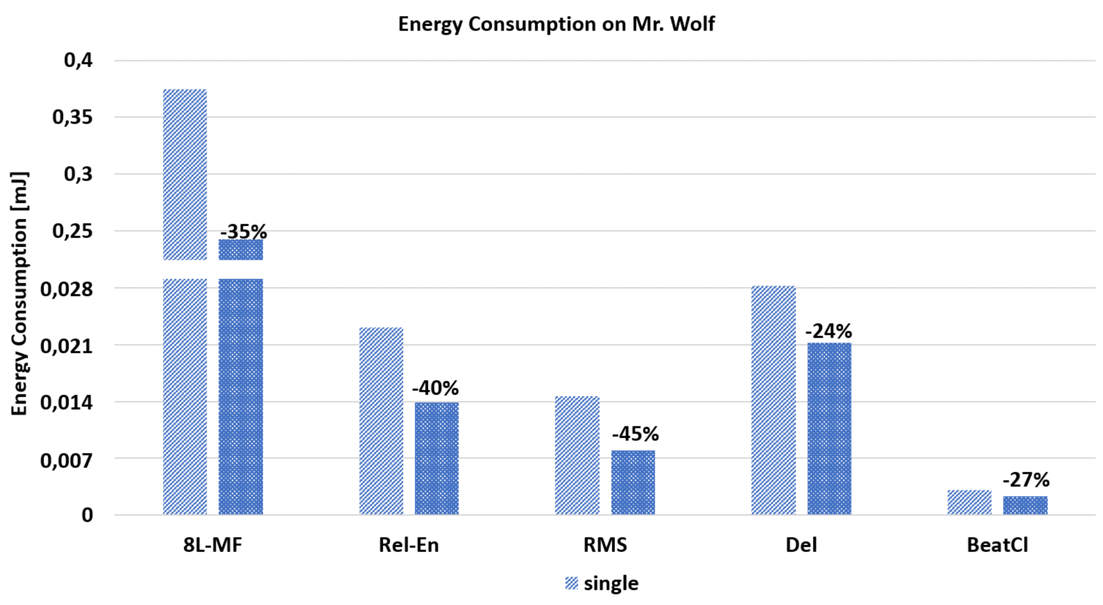


Figure 5.5: Energy consumption in the ECG modules.

Furthermore, the Relative Energy (Rel-En) has a speed-up of 7.0× with an energy saving of 40%. The parallelization is performed dividing the acquired window into a number of chunks equal to the number of cores in the cluster. In this way, it is possible to assign a chunk of data to each core with a small overlap between chunks. The loss in speed-up in this case derives from the overlap between chunks, which ensures the continuity of the signal.

The RMS demonstrate an higher predisposition to parallel computing, leading to a nearly ideal speed-up with a data level parallelization. The speed-up is around 7.7× with an energy saving of 45%. The last two kernels, delineation (Del) and beat classifier (BeatCl), returns a contained speed-ups, 5.0× and 5.7× respectively. The main reason for this results is that the parallelism is based on the number of heart beats contained in the considered window. Hence, if the number of the heart beats is not a multiple of the number of cores involved in the elaboration, the execution suffers of workload unbalanced. Otherwise, in the cases on which the number of beats is a multiple of the number of cores in the cluster, the obtained speed-ups is nearly ideal. Thus, the energy gain is equal to 24% and 27% respectively, passing from the FC to the CL execution.

5.2.6 Future Work

Currently, the targeting of ECG signals on PULP platform is at a preliminary stage. There are additional promising solutions that are worth to be explored to further improve the performance (latency and/or energy consumption). One possibility is to include in the PULP platform, supplementary hardware components. For instance, the development and the integration of specialized hardware (i.e., accelerators) to speed-up the computation of the most complex kernels (or just the most frequent operations) leads to improve the energy efficiency of the system. The integration of a non-general purpose accelerator produces beneficial effects only to a specific part of the application, leading to an inevitable loss in flexibility. Another promising direction is to integrate a Coarse-Grained Reconfigurable Arrays (CGRA), significantly improving the overall performance and energy efficiency of the system, maintaining a high degree of flexibility.

As demonstrated in the previous case studies, the PULP platform aims to satisfy heavy computation requirements, maintaining a high level of power efficiency. ECG signal processing does not require an intense workload, thus the current versions of the PULP platform result oversized for this kind of task. Due to the limited workload, usually constituted by concatenated *if* conditions, and the small amount of data to elaborate, the cluster is for almost the totality of the time in sleep mode, waiting for a new chunk of data to process. For this reason, the requirements of the architecture can be relaxed to further improve energy efficiency. For instance, the memory size can be reduced, limiting the static power contribution of the leakage. Moreover, the cluster can be re-designed, including a different version of the cores, more suitable for this processing.

Chapter 6

Conclusion

This dissertation explores Human Machine Interfaces (HMIs) in different biomedical scenarios, addressing typical challenges in wearable and implantable devices for diagnostic, monitoring and prosthetic purposes, and suggesting a methodology for the tailoring of such applications on cutting edge embedded architectures. The entire work can be considered as a guideline for the development of optimized algorithms for the processing of different kinds of biosignals, showing case studies on different scenarios to prove the effectiveness of the suggested approaches. The biosignal considered in this thesis are EMG, EEG and ECG. The thesis exploits a complete framework for EMG-based gesture recognition, including a novel algorithm for fast online training. It presents a simple and scalable solution for supervised pattern recognition, leveraging the hardware and software co-design, which ranges from algorithm to embedded optimization. This solution has been tested on 10 subjects in a typical gesture recognition scenario. HD computing with online learning reaches 85% accuracy on 11 gestures recognition, which is aligned with the SoA. Furthermore, by virtue of the efficient Mr. Wolf multi-core processor, the energy budget required to run the learning part with 11 gestures is 10.04mJ, and 83.2μJ for one classification. The system works to an average power of 10.4mW in classification, ensuring around 29h of autonomy with a battery of 100mAh.

The second signal (EEG) has multiple applications in the monitoring and the detection of certain behaviors or pathology of the brain. For instance, to evaluate the brain respond to external stimuli or the state of neurological disorders such as seizure detection. The thesis describes a machine learning-based, automated, embedded, and real-time EEG monitoring system, targeting the analysis of the frequency tagging response in a wide range of frequency stimulation including both the standard range of SSVEPs (> 6 Hz) and the low-frequency range related to slower, higher-order neural processing (0.5-6 Hz). Moreover, this approach is compared with a standard method that implies manual inspection of the traces and compute-intensive algorithms, demonstrating that the proposed approach is suitable for a real-time parallel ultra-low power implementations, within a power budget of 56mW (8-cores) and maintaining a high accuracy in detection (92%, in line with the SoA). Furthermore, the thesis presents a seizure detection implementation, showing that tailoring the execution on the PULP platform leads to lower execution time and power budget with respect to other commercial solutions, 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 using eight cores with respect to commercial MCUs, leads to 26× energy gain (more than 96% less) with respect to Ambiq Apollo and up to 140× compared to STM32F427x (99.30% energy savings).

Regarding the ECG signals, the dissertation explores a collection of typical kernels that can be used for the development of a complete ECG-based system form monitoring the heart electrical activity. In particular, these kernels are used for the filtering, the enhancement, and the delineation of the signals, and the inference of significant features. To test all the implementations, data from well known on-line databases (i.e. QTDB and MITDB) are used. Moreover, the execution of these kernels is optimized by splitting the workload among the 8-cores inside the cluster of Mr. Wolf. Some of these kernels demonstrate a great predisposition to the parallel execution, while others suffer from the overhead or the workload unbalanced derived by the parallelization.

With the fast evolution of technology, the processors are becoming increasingly efficient and powerful. Several doors are still open in hardware and software optimization to further improve their performance and energy efficiency. For instance, changing the technology used to develop the processors (reducing the form factor) would lead to an improvement in energy efficiency, reducing the leakage. This solution is useful for applications that do not require high computational power or demonstrate a meager predisposition to parallel computing (i.e. ECG application). A different approach is to increase the number of processors in the cluster (or the number of clusters in the SoC) to handle applications that require a conspicuous computational power, but demonstrating a high predisposition for parallel computing. Moreover, the development and the inclusion of specialized hardware for the acceleration of common kernels in the architectures is fundamental to achieve considerable improvements in performance, but still leaving the flexibility of a fully programmable architecture. Finally, the exploration of biosignals can be extended to other types of scenarios and applications in biomedical, pursuing the aim of developing more advanced unobtrusive embedded devices for biosignal processing.

Bibliography

[1] Jun Liu, Fan Zhang, and He Helen Huang. An open and configurable embedded system for emg pattern recognition implementation for artificial arms. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 4095–4098. IEEE, 2014.

[2] Xiaorong Zhang, He Huang, and Qing Yang. Real-time implementation of a self-recovery emg pattern recognition interface for artificial arms. In *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pages 5926–5929. IEEE, 2013.

[3] Simone Benatti, Filippo Casamassima, Bojan Milosevic, Elisabetta Farella, Philipp Schönle, Schekeb Fateh, Thomas Burger, Qiuting Huang, and Luca Benini. A versatile embedded platform for emg acquisition and gesture recognition. *IEEE Transactions on Biomedical Circuits and Systems*, 9(5):620–630, 2015.

[4] Matteo Rossi, Simone Benatti, Elisabetta Farella, and Luca Benini. Hybrid emg classifier based on hmm and svm for hand gesture recognition in prosthetics. In *Industrial Technology (ICIT), 2015 IEEE International Conference on*, pages 1700–1705. IEEE, 2015.

[5] Benatti et al. Scalable eeg seizure detection on an ultra low power multi-core architecture. In *Biomedical Circuits and Systems Conference (BioCAS), 2016 IEEE*, pages 86–89. IEEE, 2016.

[6] Victor Parsonnet, Seymour Furman, and Nicholas PD Smyth. Implantable cardiac pacemakers status report and resource guideline: Pacemaker study group. *Circulation*, 50(4): page5–A, 1974.

[7] Medtronic REVEAL. `http://www.medtronicdiagnostics.com`, 2019.

[8] Medtronic DBS. `http://www.medtronicdiagnostics.com`, 2019.

[9] Neurospace. `http://www.neuropace.com/`.

[10] Marco Capogrosso, Tomislav Milekovic, David Borton, Fabien Wagner, Eduardo Martin Moraud, Jean-Baptiste Mignardot, Nicolas Buse, Jerome Gandar, Quentin Barraud,

David Xing, et al. A brain–spine interface alleviating gait deficits after spinal cord injury in primates. *Nature*, 539(7628):284, 2016.

[11] Ali Moin, George Alexandrov, Benjamin C Johnson, Igor Izyumin, Fred Burghardt, Kedar Shah, Sat Pannu, Elad Alon, Rikky Muller, and Jan M Rabaey. Powering and communication for omni: a distributed and modular closed-loop neuromodulation device. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4471–4474. IEEE, 2016.

[12] Felice T Sun and Martha J Morrell. Closed-loop neurostimulation: the clinical experience. *Neurotherapeutics*, 11(3):553–563, 2014.

[13] Victor Kartsch, Simone Benatti, Davide Rossi, and Luca Benini. A wearable eeg-based drowsiness detection system with blink duration and alpha waves analysis. In *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 251–254. IEEE, 2017.

[14] Alice B Brandwein, John J Foxe, John S Butler, Hans-Peter Frey, Juliana C Bates, Lisa H Shulman, and Sophie Molholm. Neurophysiological indices of atypical auditory processing and multisensory integration are associated with symptom severity in autism. *Journal of autism and developmental disorders*, 45(1):230–244, 2015.

[15] Dongjin Seo, Ryan M Neely, Konlin Shen, Utkarsh Singhal, Elad Alon, Jan M Rabaey, Jose M Carmena, and Michel M Maharbiz. Wireless recording in the peripheral nervous system with ultrasonic neural dust. *Neuron*, 91(3):529–539, 2016.

[16] Simone Benatti, Bojan Milosevic, Elisabetta Farella, Emanuele Gruppioni, and Luca Benini. A prosthetic hand body area controller based on efficient pattern recognition control strategies. *Sensors*, 17(4):869, 2017.

[17] Lauren H Smith, Levi J Hargrove, Blair A Lock, and Todd A Kuiken. Determining the optimal window length for pattern recognition-based myoelectric control: balancing the competing effects of classification error and controller delay. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(2):186–192, 2010.

[18] Jeremy Constantin, Ahmed Dogan, Oskar Andersson, Pascal Meinerzhagen, Joachim Neves Rodrigues, David Atienza, and Andreas Burg. Tamarisc-cs: An ultra-low-power application-specific processor for compressed sensing. In *2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, pages 159–164. IEEE, 2012.

[19] Fabio Montagna, Simone Benatti, and Davide Rossi. Flexible, scalable and energy efficient bio-signals processing on the pulp platform: A case study on seizure detection. *Journal of Low Power Electronics and Applications*, 7(2):16, 2017.

[20] Marco Tomasini, Simone Benatti, Bojan Milosevic, Elisabetta Farella, and Luca Benini. Power line interference removal for high-quality continuous biosignal monitoring with low-power wearable devices. *IEEE Sensors Journal*, 16(10):3887–3895, 2016.

[21] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[22] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on pattern analysis and machine intelligence*, 20(12):1371–1375, 1998.

[23] T Scott Saponas, Desney S Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A Landay. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 167–176. ACM, 2009.

[24] touch bionics. `http://www.touchbionics.com/products`, 2018.

[25] Ottobock. `https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/myoelectric-prosthetics/`, 2018.

[26] R. Meattini, S. Benatti, U. Scarcia, D. De Gregorio, L. Benini, and C. Melchiorri. An semg-based humanrobot interface for robotic hands using machine learning and synergies. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 8 (7):1149–1158, July 2018. ISSN 2156-3950. doi: 10.1109/TCPMT.2018.2799987.

[27] Haoshi Zhang, Yaonan Zhao, Fuan Yao, Lisheng Xu, Peng Shang, and Guanglin Li. An adaptation strategy of using lda classifier for emg pattern recognition. In *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pages 4267–4270. IEEE, 2013.

[28] M. R. Ahsan, M. I. Ibrahimy, and O. O. Khalifa. Electromygraphy (emg) signal based hand gesture recognition using artificial neural network (ann). In *2011 4th International Conference on Mechatronics (ICOM)*, pages 1–6, May 2011. doi: 10.1109/ICOM.2011.5937135.

[29] Mohammadreza Asghari Oskoei, Huosheng Hu, et al. Support vector machine-based classification scheme for myoelectric control applied to upper limb. *IEEE Trans. Biomed. Engineering*, 55(8):1956–1965, 2008.

[30] P. Zhang and J. Peng. Svm vs regularized least squares classification. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 176–179 Vol.1, Aug 2004. doi: 10.1109/ICPR.2004.1334050.

[31] A. D. I. Falih, W. A. Dharma, and S. Sumpeno. Classification of emg signals from forearm muscles as automatic control using naive bayes. In *2017 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 346–351, Aug 2017. doi: 10.1109/ISITIA.2017.8124107.

[32] D. Farina and A. Holobar. Characterization of human motor units from surface emg decomposition. *Proceedings of the IEEE*, 104(2):353–373, 2016.

[33] Manfredo Atzori, Matteo Cognolato, and Henning Müller. Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in neurorobotics*, 10:9, 2016.

[34] Asim Waris, Irene Mendez, Kevin Englehart, Winnie Jensen, and Ernest Nlandu Kamavuako. On the robustness of real-time myoelectric control investigations: A multiday fitts law approach. *Journal of Neural Engineering*, 2018.

[35] A. J. Ishak, S. A. Ahmad, A. C. Soh, N. A. Naraina, R. M. R. Jusoh, and W. Chikamune. Design of a wireless surface emg acquisition system. In *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 1–6, Nov 2017.

[36] Bojan Milosevic, Elisabetta Farella, and Simone Benatti. Exploring arm posture and temporal variability in myoelectric hand gesture recognition. In *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, pages 1032–1037. IEEE, 2018.

[37] Asim Waris, Imran Khan Niazi, Mohsin Jamil, Kevin Englehart, Winnie Jensen, and Ernest Nlandu Kamavuako. Multiday evaluation of techniques for emg based classification of hand motions. *IEEE journal of biomedical and health informatics*, 2018.

[38] Z Zainuddin, N Mahat, and Y Abu Hassan. Improving the convergence of the backpropagation algorithm using local adaptive techniques. In *International Conference on Computational Intelligence*, pages 173–176. Citeseer, 2004.

[39] Simone Benatti, Giovanni Rovere, Jonathan Bösser, Fabio Montagna, Elisabetta Farella, Horian Glaser, Philipp Schönle, Thomas Burger, Schekeb Fateh, Qiuting Huang, et al. A sub-10mw real-time implementation for emg hand gesture recognition based on a multicore biomedical soc. In *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 139–144. IEEE, 2017.

[40] United Nations. `http://www.un.org/`, 2016.

[41] Brain Facts. `http://www.brainfacts.org/`, 2016.

[42] Ernst Niedermeyer and FH Lopes da Silva. *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005.

[43] D Puthankattil Subha, Paul K Joseph, Rajendra Acharya, and Choo Min Lim. Eeg signal analysis: a survey. *Journal of medical systems*, 34(2):195–212, 2010.

[44] Trevor Thompson, Tony Steffert, Tomas Ros, Joseph Leach, and John Gruzelier. Eeg applications for sport and performance. *Methods*, 45(4):279–288, 2008.

[45] Narayanan Srinivasan. Cognitive neuroscience of creativity: Eeg based approaches. *Methods*, 42(1):109–116, 2007.

[46] Steven J Luck. *An introduction to the event-related potential technique*. MIT press, 2014.

[47] Terence W Picton, M Sasha John, Andrew Dimitrijevic, and David Purcell. Human auditory steady-state responses: Respuestas auditivas de estado estable en humanos. *International journal of audiology*, 42(4):177–219, 2003.

[48] Anthony M Norcia, L Gregory Appelbaum, Justin M Ales, Benoit R Cottereau, and Bruno Rossion. The steady-state visual evoked potential in vision research: a review. *Journal of vision*, 15(6):4–4, 2015.

[49] Guangyu Bin, Xiaorong Gao, Zheng Yan, Bo Hong, and Shangkai Gao. An online multi-channel ssvep-based brain–computer interface using a canonical correlation analysis method. *Journal of neural engineering*, 6(4):046002, 2009.

[50] Yee Joon Kim, Marcia Grabowecky, Ken A Paller, Krishnakumar Muthu, and Satoru Suzuki. Attention induces synchronization-based response gain in steady-state visual evoked potentials. *Nature neuroscience*, 10(1):117–125, 2007.

[51] Marco Buiatti, Marcela Peña, and Ghislaine Dehaene-Lambertz. Investigating the neural correlates of continuous speech computation with frequency-tagged neuroelectric responses. *Neuroimage*, 44(2):509–519, 2009.

[52] Bruno Rossion, Katrien Torfs, Corentin Jacques, and Joan Liu-Shuang. Fast periodic presentation of natural images reveals a robust face-selective electrophysiological response in the human brain. *Journal of vision*, 15(1):18–18, 2015.

[53] C Kabdebon, M Pena, M Buiatti, and G Dehaene-Lambertz. Electrophysiological evidence of statistical learning of long-distance dependencies in 8-month-old preterm and full-term infants. *Brain and language*, 148:25–36, 2015.

[54] Adélaïde de Heering and Bruno Rossion. Rapid categorization of natural face images in the infant right hemisphere. *Elife*, 4:e06564, 2015.

[55] S Benatti, B Milosevic, M Tomasini, E Farella, P Schönle, P Bunjaku, G Rovere, S Fateh, Q Huang, and L Benini. Multiple biopotentials acquisition system for wearable applications. *Proc. of SmartMedDev*, 2015.

[56] Raffaele Gravina, Parastoo Alinia, Hassan Ghasemzadeh, and Giancarlo Fortino. Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. *Information Fusion*, 35:68–80, 2017.

[57] Giancarlo Fortino, Roberta Giannantonio, Raffaele Gravina, Philip Kuryloski, and Roozbeh Jafari. Enabling effective programming and flexible management of efficient body sensor network applications. *IEEE Transactions on Human-Machine Systems*, 43 (1):115–133, 2013.

[58] Faisal A Al-Otaibi, Clement Hamani, and Andres M Lozano. Neuromodulation in epilepsy. *Neurosurgery*, 69(4):957–979, 2011.

[59] George Nune, Christopher DeGiorgio, and Christianne Heck. Neuromodulation in the treatment of epilepsy. *Current treatment options in neurology*, 17(10):43, 2015.

[60] U Rajendra Acharya, S Vinitha Sree, Ang Peng Chuan Alvin, and Jasjit S Suri. Use of principal component analysis for automatic classification of epileptic eeg activities in wavelet framework. *Expert Systems with Applications*, 39(10):9072–9078, 2012.

[61] Abdulhamit Subasi and M Ismail Gursoy. Eeg signal classification using pca, ica, lda and support vector machines. *Expert Systems with Applications*, 37(12):8659–8666, 2010.

[62] Chunmei Wang, Junzhong Zou, Jian Zhang, Min Wang, and Rubin Wang. Feature extraction and recognition of epileptiform activity in eeg by combining pca with apen. *Cognitive neurodynamics*, 4(3):233–240, 2010.

[63] Medtronic. `http://www.medtronic.com/`.

[64] J. Yoo, L. Yan, D. El-Damak, M. B. Altaf, A. Shoeb, H. J. Yoo, and A. Chandrakasan. An 8-channel scalable eeg acquisition soc with fully integrated patient-specific seizure classification and recording processor. In *2012 IEEE International Solid-State Circuits Conference*, pages 292–294, Feb 2012. doi: 10.1109/ISSCC.2012.6177019.

[65] Kyong Ho Lee and Naveen Verma. A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *IEEE Journal of Solid-State Circuits*, 48(7):1625–1637, 2013.

[66] Amey Kulkarni et al. Sketching-based high-performance biomedical big data processing accelerator. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pages 1138–1141. IEEE, 2016.

[67] Davide Rossi, Igor Loi, Germain Haugou, and Luca Benini. Ultra-low-latency lightweight dma for tightly coupled multi-core clusters. In *Proceedings of the 11th ACM Conference on Computing Frontiers*, CF '14, pages 15:1–15:10, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2870-8. doi: 10.1145/2597917.2597922. URL `http://doi.acm.org/10.1145/2597917.2597922`.

[68] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini. Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1201–1206, March 2016.

[69] D. Rossi. Sub-pj per operation scalable computing: The pulp experience. In *2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pages 1–3, Oct 2016. doi: 10.1109/S3S.2016.7804389.

[70] Michael Gautschi, Davide Rossi, and Luca Benini. Customizing an open source processor to fit in an ultra-low power cluster with a shared l1 memory. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, GLSVLSI '14, pages 87–88, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2816-6. doi: 10.1145/2591513.2591569. URL `http://doi.acm.org/10.1145/2591513.2591569`.

[71] M. Gautschi, M. Schaffner, F. K. Gürkaynak, and L. Benini. 4.6 a 65nm 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 *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 82–83, Jan 2016. doi: 10.1109/ISSCC.2016.7417917.

[72] Igor Loi, Davide Rossi, Germain Haugou, Michael Gautschi, and Luca Benini. Exploring multi-banked shared-l1 program cache on ultra-low power, tightly coupled processor clusters. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, CF '15, pages 64:1–64:8, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3358-0. doi: 10.1145/2742854.2747288. URL `http://doi.acm.org/10.1145/2742854.2747288`.

[73] Adam Teman, Davide Rossi, Pascal Meinerzhagen, Luca Benini, and Andreas Burg. Power, area, and performance optimization of standard cell memory arrays through controlled placement. *ACM Trans. Des. Autom. Electron. Syst.*, 21(4):59:1–59:25, may 2016. ISSN 1084-4309. doi: 10.1145/2890498. URL `http://doi.acm.org/10.1145/2890498`.

[74] Antonio Pullini, Davide Rossi, Igor Loi, Giuseppe Tagliavini, and Luca Benini. Mr. wolf: An energy-precision scalable parallel ultra low power soc for iot edge processing. *IEEE Journal of Solid-State Circuits*, 2019.

[75] P. Davide Schiavone et al. Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications. In *PATMOS*, pages 1–8, Sept 2017. doi: 10.1109/PATMOS.2017.8106976.

[76] Rohit Chandra, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[77] A. Marongiu and L. Benini. An openmp compiler for efficient use of distributed scratchpad memory in mpsocs. *IEEE Transactions on Computers*, 61(2):222–236, Feb 2012. ISSN 0018-9340. doi: 10.1109/TC.2010.199.

[78] A. Moin, A. Zhou, A. Rahimi, S. Benatti, A. Menon, S. Tamakloe, J. Ting, N. Yamamoto, Y. Khan, F. Burghardt, L. Benini, A. C. Arias, and J. M. Rabaey. An emg gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2018. doi: 10.1109/ISCAS.2018.8351613.

[79] Sidharth Pancholi and Amit M Joshi. Portable emg data acquisition module for upper limb prosthesis application. *IEEE Sensors Journal*, 18(8):3436–3443, 2018.

[80] Marie-Françoise Lucas, Adrien Gaufriau, Sylvain Pascual, Christian Doncarli, and Dario Farina. Multi-channel surface emg classification using support vector machines and signal-based wavelet optimization. *Biomedical Signal Processing and Control*, 3(2):169–174, 2008.

[81] Sebastian Bitzer and Patrick Van Der Smagt. Learning emg control of a robotic hand: towards active prostheses. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2819–2823. IEEE, 2006.

[82] Ahmet Alkan and Mücahid Günay. Identification of emg signals using discriminant analysis and svm classifier. *Expert Systems with Applications*, 39(1):44–47, 2012.

[83] Claudio Castellini, Emanuele Gruppioni, Angelo Davalli, and Giulio Sandini. Fine detection of grasp force and posture by amputees via surface electromyography. *Journal of Physiology-Paris*, 103(3-5):255–262, 2009.

[84] Claudio Castellini and Patrick van der Smagt. Surface emg in advanced hand prosthetics. *Biological cybernetics*, 100(1):35–47, 2009.

[85] B.E. Boser *et al.* A training algorithm for optimal margin classifiers. In *Proc. of the Workshop on Computational learning theory*, pages 144–152, 1992.

[86] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[87] Halima Bensmail and Gilles Celeux. Regularized gaussian discriminant analysis through eigenvalue decomposition. *Journal of the American statistical Association*, 91(436): 1743–1748, 1996.

[88] Paolo Gentile, Marco Pessione, Antonio Suppa, Alessandro Zampogna, and Fernanda Irrera. Embedded wearable integrating real-time processing of electromyography signals. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 1, page 600, 2017.

[89] Xilin Liu, Jacob Sacks, Milin Zhang, Andrew G Richardson, Timothy H Lucas, and Jan Van der Spiegel. The virtual trackpad: An electromyography-based, wireless, real-time, low-power, embedded hand-gesture-recognition system using an event-driven artificial neural network. *IEEE Trans. Circuits Syst. II Express Briefs*, 64:1257–1261, 2017.

[90] Yu Hu, Yongkang Wong, Wentao Wei, Yu Du, Mohan Kankanhalli, and Weidong Geng. A novel attention-based hybrid cnn-rnn architecture for semg-based gesture recognition. *PloS one*, 13(10):e0206049, 2018.

[91] Janne M Hahne, F Biessmann, Ning Jiang, H Rehbaum, Dario Farina, FC Meinecke, K-R Müller, and LC Parra. Linear and nonlinear regression techniques for simultaneous and proportional myoelectric control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(2):269–279, 2014.

[92] Ning Jiang, Kevin B Englehart, and Philip A Parker. Extracting simultaneous and proportional neural control information for multiple-dof prostheses from the surface electromyographic signal. *IEEE transactions on Biomedical Engineering*, 56(4):1070–1080, 2009.

[93] Kevin R Wheeler, Mindy H Chang, and Kevin H Knuth. Gesture-based control and emg decomposition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(4):503–514, 2006.

[94] ARM Cortex M4. `https://developer.arm.com/products/processors/cortex-m/cortex-m4`, 2013.

[95] OMAP processor. `http://www.ti.com`, 2013.

[96] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009. ISSN 1866-9956. doi: 10.1007/s12559-009-9009-8. URL `http://dx.doi.org/10.1007/s12559-009-9009-8`.

[97] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey. Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals. *Proceedings of the IEEE*, pages 1–21, 2018. ISSN 0018-9219. doi: 10.1109/JPROC.2018.2871163.

[98] Bernhard Scholkopf, Kah-Kay Sung, Christopher JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997.

[99] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[100] Pentti Kanerva. *Sparse Distributed Memory*. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262111322.

[101] Abbas Rahimi, Simone Benatti, Pentti Kanerva, Luca Benini, and Jan M. Rabaey. Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In *IEEE International Conference on Rebooting Computing*, October 2016.

[102] Fabio Montagna, Abbas Rahimi, Simone Benatti, Davide Rossi, and Luca Benini. Pulp-hd: Accelerating brain-inspired high-dimensional computing on a parallel ultra-low power platform. In *Proceedings of the 55th Annual Design Automation Conference*, page 111. ACM, 2018.

[103] Abbas Rahimi, Pentti Kanerva, José del R Millán, and Jan M. Rabaey. Hyperdimensional computing for noninvasive brain–computer interfaces: Blind and one-shot classification of EEG error-related potentials. *10th ACM/EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, 3 2017. doi: 10.4108/eai.22-3-2017.152397.

[104] Abbas Rahimi, Artiom Tchouprina, Pentti Kanerva, José del R Millán, and Jan M Rabaey. Hyperdimensional computing for blind and one-shot classification of eeg error-related potentials. *Mobile Networks and Applications*, pages 1–12, Oct 2017.

[105] A. Burrello, K. Schindler, L. Benini, and A. Rahimi. One-shot learning for iEEG seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing. In *Biomedical Circuits and Systems Conference (BioCAS), 2018 IEEE*, pages 1–4, 2018.

[106] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Åekercioglu. Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE Transactions on Neural Networks and Learning Systems*, 28(6):1250–1262, June 2017. ISSN 2162-237X. doi: 10.1109/TNNLS.2016.2535338.

[107] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1 (2):139–159, 2009.

[108] D. Rossi et al. A self-aware architecture for pvt compensation and power nap in near threshold processors. *IEEE Design Test*, 34(6):46–53, Dec 2017. ISSN 2168-2356.

[109] J-U Chu, Inhyuk Moon, and M-S Mun. A real-time emg pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand. *IEEE Transactions on biomedical engineering*, 53(11):2232–2239, 2006.

[110] Mahdi Khezri and Mehran Jahed. Real-time intelligent pattern recognition algorithm for surface emg signals. *Biomedical engineering online*, 6(1):45, 2007.

[111] David Bellasi, Philipp Schönle, Qiuting Huang, and Luca Benini. A wide tuning-range adfll for mw-socs with dithering-enhanced accuracy in 65 nm cmos. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[112] Mattia Salvaro, Simone Benatti, Victor J Kartsch, Marco Guermandi, and Luca Benini. A minimally invasive low-power platform for real-time brain computer interaction based on canonical correlation analysis. *IEEE Internet of Things Journal*, 6(1):967–977, 2018.

[113] Giulia Barbati, Camillo Porcaro, Filippo Zappasodi, Paolo Maria Rossini, and Franca Tecchio. Optimization of an independent component analysis approach for artifact identification and removal in magnetoencephalographic signals. *Clinical Neurophysiology*, 115(5):1220 – 1232, 2004. ISSN 1388-2457.

[114] Arnaud Delorme, Terrence Sejnowski, and Scott Makeig. Enhanced detection of artifacts in eeg data using higher-order statistics and independent component analysis. *Neuroimage*, 34(4):1443–1449, 2007.

[115] D. Mantini, R. Franciotti, G.L. Romani, and V. Pizzella. Improving {MEG} source localizations: An automated method for complete artifact removal based on independent component analysis. *NeuroImage*, 40(1):160 – 173, 2008. ISSN 1053-8119.

[116] Yandong Li, Zhongwei Ma, Wenkai Lu, and Yanda Li. Automatic removal of the eye blink artifact from eeg using an ica-based template matching approach. *Physiological measurement*, 27(4):425, 2006.

[117] Filipa Campos Viola, Jeremy Thorne, Barrie Edmonds, Till Schneider, Tom Eichele, and Stefan Debener. Semi-automatic identification of independent components representing eeg artifact. *Clinical Neurophysiology*, 120(5):868 – 877, 2009. ISSN 1388-2457.

[118] Carrie A. Joyce, Irina F. Gorodnitsky, and Marta Kutas. Automatic removal of eye movement and blink artifacts from eeg data using blind component separation. *Psychophysiology*, 41(2):313–325, 2004. ISSN 1469-8986.

[119] Y Okada, J Jung, and T Kobayashi. An automatic identification and removal method for eye-blink artifacts in event-related magnetoencephalographic measurements. *Physiological measurement*, 28(12):1523, 2007.

[120] Andrea Mognon, Jorge Jovicich, Lorenzo Bruzzone, and Marco Buiatti. Adjust: An automatic eeg artifact detector based on the joint use of spatial and temporal features. *Psychophysiology*, 48(2):229–240, 2011. ISSN 1469-8986.

[121] Nai Ding, Lucia Melloni, Hang Zhang, Xing Tian, and David Poeppel. Cortical tracking of hierarchical linguistic structures in connected speech. *Nature neuroscience*, 19(1): 158–164, 2016.

[122] Daniel Baldauf and Robert Desimone. Neural mechanisms of object-based attention. *Science*, 344(6182):424–427, 2014.

[123] Ali Bashashati, Mehrdad Fatourechi, Rabab K Ward, and Gary E Birch. A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. *Journal of Neural engineering*, 4(2):R32, 2007.

[124] S. G. Mason and G. E. Birch. A general framework for brain-computer interface design. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(1):70–85, March 2003. ISSN 1534-4320. doi: 10.1109/TNSRE.2003.810426.

[125] Vojkan Mihajlović, Bernard Grundlehner, Ruud Vullers, and Julien Penders. Wearable, wireless eeg solutions in daily life applications: what are we missing? *IEEE journal of biomedical and health informatics*, 19(1):6–21, 2014.

[126] Arnaud Delorme and Scott Makeig. Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis. *J. Neurosci. Methods*, page 21.

[127] Yann Renard, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand, and Anatole Lécuyer. OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments. *Presence: Teleoperators and Virtual Environments / Presence Teleoperators and Virtual Environments*, 19(1):35–53, 2010.

[128] R. Mahajan, C. A. Majmudar, S. Khatun, B. I. Morshed, and G. M. Bidelman. Neuromonitor ambulatory eeg device: Comparative analysis and its application for cognitive load assessment. In *2014 IEEE Healthcare Innovation Conference (HIC)*, pages 133–136, Oct 2014. doi: 10.1109/HIC.2014.7038892.

[129] Quasar. `http://www.quasarusa.com`, 2013.

[130] S. Patki, B. Grundlehner, A. Verwegen, S. Mitra, J. Xu, A. Matsumoto, R. F. Yazicioglu, and J. Penders. Wireless eeg system with real time impedance monitoring and active electrodes. In *2012 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 108–111, Nov 2012. doi: 10.1109/BioCAS.2012.6418408.

[131] Emotiv. `http://emotiv.com`, 2015.

[132] NeuroSky. `http://www.neurosky`, 2016.

[133] Gtec. `http://www.gtec.at`, 2012.

[134] OpernBCI. `http://openbci.com`, 2014.

[135] D. Giovanelli, B. Milosevic, and E. Farella. Bluetooth low energy for data streaming: Application-level analysis and recommendation. In *2015 6th International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 216–221, June 2015. doi: 10.1109/IWASI.2015.7184945.

[136] D. Brunelli, E. Farella, D. Giovanelli, B. Milosevic, and I. Minakov. Design considerations for wireless acquisition of multichannel semg signals in prosthetic hand control. *IEEE Sensors Journal*, 16(23):8338–8347, Dec 2016. ISSN 1530-437X. doi: 10.1109/JSEN.2016.2596712.

[137] Davide Rossi, Antonio Pullini, Igor Loi, Michael Gautschi, Frank K. Gurkaynak, Andrea Bartolini, Philippe Flatresse, and Luca Benini. A 60 gops/w, -1.8 v to 0.9 v body bias ulp cluster in 28nm utbb fd-soi technology. *Solid-State Electronics*, 117:170 – 184, 2016. ISSN 0038-1101.

[138] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gurkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beigné, F. Clermidy, F. Abouzeid, P. Flatresse, and L. Benini. 193 mops/mw @ 162 mops, 0.32v to 1.15v voltage range multi-core accelerator for energy efficient parallel and sequential digital processing. In *2016 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS XIX)*, pages 1–3, April 2016. doi: 10.1109/CoolChips.2016.7503670.

[139] D. Regan. *Human brain electrophysiology: evoked potentials and evoked magnetic fields in science and medicine*. Elsevier, 1989. ISBN 9780444013248.

[140] George Klem et al. The ten-twenty electrode system of the international federation. *Electroencephalogr Clin Neurophysiol*, 1999.

[141] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[142] Turkey N Alotaiby, Saleh A Alshebeili, Tariq Alshawi, Ishtiaq Ahmad, and Fathi E Abd El-Samie. Eeg seizure detection and prediction algorithms: a survey. *EURASIP Journal on Advances in Signal Processing*, 2014(1):1, 2014.

[143] S Benatti, F Montagna, D Rossi, and L Benini. Scalable eeg seizure detection on an ultra low power multi-core architecture. In *Biomedical Circuits and Systems Conference (BioCAS), 2016 IEEE*, pages 86–89. IEEE, 2016.

[144] David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

[145] Hui Chen, Jing Zhang, Yan Xu, Budong Chen, and Kuan Zhang. Performance comparison of artificial neural network and logistic regression model for differentiating lung nodules on ct scans. *Expert Systems with Applications*, 39(13):11503–11509, 2012.

[146] Dursun Delen, Glenn Walker, and Amit Kadam. Predicting breast cancer survivability: a comparison of three data mining methods. *Artificial intelligence in medicine*, 34(2): 113–127, 2005.

[147] Behshad Hosseinifard, Mohammad Hassan Moradi, and Reza Rostami. Classifying depression patients and normal subjects using machine learning techniques and nonlinear features from eeg signal. *Computer methods and programs in biomedicine*, 109(3):339–345, 2013.

[148] Wei-Ming Chen, Herming Chiueh, Tsan-Jieh Chen, Chia-Lun Ho, Chi Jeng, Ming-Dou Ker, Chun-Yu Lin, Ya-Chun Huang, Chia-Wei Chou, Tsun-Yuan Fan, et al. A fully integrated 8-channel closed-loop neural-prosthetic cmos soc for real-time epileptic seizure control. *IEEE journal of solid-state circuits*, 49(1):232–247, 2014.

[149] Kunjan Patel, Chern-Pin Chua, Stephen Fau, and Chris J Bleakley. Low power real-time seizure detection for ambulatory eeg. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1–7. IEEE, 2009.

[150] Naveen Verma, Ali Shoeb, Jose Bohorquez, Joel Dawson, John Guttag, and Anantha P Chandrakasan. A micro-power eeg acquisition soc with integrated feature extraction processor for a chronic seizure detection system. *IEEE Journal of Solid-State Circuits*, 45 (4):804–816, 2010.

[151] Monty A Escabí, Heather L Read, Jonathan Viventi, Dae-Hyeong Kim, Nathan C Higgins, Douglas A Storace, Andrew SK Liu, Adam M Gifford, John F Burke, Matthew Campisi, et al. A high-density, high-channel count, multiplexed μecog array for auditory-cortex recordings. *Journal of neurophysiology*, 112(6):1566–1583, 2014.

[152] J. Kwong and A. P. Chandrakasan. An energy-efficient biomedical signal processing platform. *IEEE Journal of Solid-State Circuits*, 46(7), July 2011. ISSN 0018-9200.

[153] J. Kwong, Y. Ramadass, N. Verma, M. Koesler, K. Huber, H. Moormann, and A. Chandrakasan. A 65nm sub-vt microcontroller with integrated sram and switched-capacitor dc-dc converter. In *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 318–616, Feb 2008. doi: 10.1109/ISSCC.2008.4523185.

[154] Jack E Volder. The cordic trigonometric computing technique. *IRE Transactions on electronic computers*, (3):330–334, 1959.

[155] John S Walther. A unified algorithm for elementary functions. In *Proceedings of the May 18-20, 1971, spring joint computer conference*, pages 379–385. ACM, 1971.

[156] M Causo, S Benatti, A Frappé, A Cathelin, E Farella, A Kaiser, L Benini, and JM Rabaey. Sampling modulation: an energy efficient novel feature extraction for biosignal processing.

[157] Yinxia Liu, Weidong Zhou, Qi Yuan, and Shuangshuang Chen. Automatic seizure detection using wavelet transform and svm in long-term intracranial eeg. *IEEE transactions on neural systems and rehabilitation engineering*, 20(6):749–755, 2012.

[158] Simone Benatti, Elisabetta Farella, and Luca Benini. Towards emg control interface for smart garments. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers: Adjunct Program*, pages 163–170. ACM, 2014.

[159] John Henry Wilkinson, Friedrich Ludwig Bauer, and C Reinsch. *Linear algebra*, volume 2. Springer, 2013.

[160] Claudio Brunelli, Fabio Campi, Claudio Mucci, Davide Rossi, Tapani Ahonen, Juha Kylliäinen, Fabio Garzia, and Jari Nurmi. Design space exploration of an open-source, ip-reusable, scalable floating-point engine for embedded applications. *Journal of Systems Architecture*, 54(12):1143 – 1154, 2008. ISSN 1383-7621. doi: http://dx.doi.org/10.1016/j.sysarc.2008.05.005. URL `http://www.sciencedirect.com/science/article/pii/S138376210800091X`.

[161] Erick L Oberstar. Fixed-point representation & fractional math. *Oberstar Consulting, revision*, 1, 2007.

[162] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[163] Ali Hossam Shoeb. *Application of machine learning to epileptic seizure onset detection and treatment*. PhD thesis, Massachusetts Institute of Technology, 2009.

[164] Suryannarayana Chandaka, Amitava Chatterjee, and Sugata Munshi. Cross-correlation aided support vector machine classifier for classification of eeg signals. *Expert Systems with Applications*, 36(2):1329–1336, 2009.

[165] U Rajendra Acharya, Filippo Molinari, S Vinitha Sree, Subhagata Chattopadhyay, Kwan-Hoong Ng, and Jasjit S Suri. Automated diagnosis of epileptic eeg using entropies. *Biomedical Signal Processing and Control*, 7(4):401–408, 2012.

[166] Nicoletta Nicolaou and Julius Georgiou. Detection of epileptic electroencephalogram based on permutation entropy and support vector machines. *Expert Systems with Applications*, 39(1):202–209, 2012.

[167] Benjamin I Rapoport, Jakub T Kedzierski, and Rahul Sarpeshkar. A glucose fuel cell for implantable brain–machine interfaces. *PloS one*, 7(6):e38436, 2012.

[168] Emmanuel J Candes et al. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

[169] Mohammed Shoaib et al. A 0.6–107 μw energy-scalable processor for directly analyzing compressively-sensed eeg. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(4):1105–1118, 2014.

[170] Venkata Rajesh Pamula et al. A 172μw compressively sampled photoplethysmographic (ppg) readout asic with heart rate estimation directly from compressively sampled data. *IEEE transactions on biomedical circuits and systems*, 11(3):487–496, 2017.

[171] Nicholas R Lomb. Least-squares frequency analysis of unequally spaced data. *Astrophysics and space science*, 39(2):447–462, 1976.

[172] Ali Shoeb et al. Application of machine learning to epileptic seizure detection. In *ICML-10*, pages 975–982, 2010.

[173] Anthony Dupre, Sarah Vincent, and Paul A Iaizzo. Basic ecg theory, recordings, and interpretation. In *Handbook of cardiac anatomy, physiology, and devices*, pages 191–201. Springer, 2005.

[174] Ruben Braojos, Giovanni Ansaloni, David Atienza, and Francisco Rincon. Embedded real-time ECG delineation methods: A comparative evaluation. In *Int. Conf. on Bioinformatics & Bioengineering (BIBE)*. IEEE, November 2012. ISBN 978-1-4673-4358-9. doi: 10.1109/BIBE.2012.6399715. URL `http://ieeexplore.ieee.org/document/6399715/`.

[175] Lara Orlandic, Elisabetta De Giovanni, Adriana Arza, Sasan Yazdani, Jean-Marc Vesin, and David Atienza. REWARD: Design, optimization, and evaluation of a real-time relative-energy wearable R-peak detection algorithm. In *Engineering in Medicine and Biology Conference (EMBC)*, July 2019.

[176] M. La Scala, G. Sblendorio, and R. Sbrizzai. Parallel-in-time implementation of transient stability simulations on a transputer network. *IEEE Trans. on Power Systems*, 9 (2):1117–1125, May 1994. ISSN 08858950. doi: 10.1109/59.317618. URL `http://ieeexplore.ieee.org/document/317618/`.

[177] Ruben Braojos, Giovanni Ansaloni, and David Atienza. A methodology for embedded classification of heartbeats using random projections. In *IEEE DATE*, May 2013. ISBN 9781467350716. doi: 10.7873/DATE.2013.189. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6513635`.

[178] Elisabetta De Giovanni, Amir Aminifar, Adrian Luca, Sasan Yazdani, Jean-Marc Vesin, and David Atienza. A patient-specific methodology for prediction of paroxysmal atrial fibrillation onset. In *Computing in Cardiology (CinC)*, September 2017. doi: 10.22489/CinC.2017.285-191. URL `http://www.cinc.org/archives/2017/pdf/285-191.pdf`.

[179] P. Laguna, R.G. Mark, A. Goldberg, and G.B. Moody. A database for evaluation of algorithms for measurement of QT and other waveform intervals in the ECG. In *Computers in Cardiology*, pages 673–676. IEEE, 1997. ISBN 0-7803-4445-6. doi: 10.1109/CIC.1997.648140. URL `http://ieeexplore.ieee.org/document/648140/`.

[180] G.B. Moody and R.G. Mark. The impact of the MIT-BIH Arrhythmia Database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001. ISSN 07395175. doi: 10.1109/51.932724. URL `http://ieeexplore.ieee.org/document/932724/`.