University of Cagliari

PhD Course in

Electronic and Computer Engineering

Class XXVI

ACQUISITION SYSTEMS AND DECODING ALGORITHMS OF PERIPHERAL NEURAL SIGNALS FOR PROSTHETIC APPLICATIONS

Scientific field

ING-INF/01 (Elettronica)

Author:	NICOLA CARTA
PhD Course Coordinator:	PROF. FABIO ROLI
Advisor:	PROF. LUIGI RAFFO

Academic year of PhD defence 2012 – 2013

Università degli Studi di Cagliari

DOTTORATO DI RICERCA

IN INGEGNERIA ELETTRONICA ED INFORMATICA

Ciclo XXVI

ACQUISITION SYSTEMS AND DECODING ALGORITHMS OF PERIPHERAL NEURAL SIGNALS FOR PROSTHETIC APPLICATIONS

Settore scientifico disciplinare di afferenza

ING-INF/01 (Elettronica)

Presentata da:	NICOLA CARTA
Coordinatore Dottorato	PROF. FABIO ROLI
Tutor/Relatore	PROF. LUIGI RAFFO

Esame finale anno accademico 2012 – 2013

Un grazie infinito alla mia famiglia ed alla mia ragazza

Abstract

During the years, neuroprosthetic applications have obtained a great deal of attention by the international research, especially in the bioengineering field, thanks to the huge investments on several proposed projects funded by the political institutions which consider the treatment of this particular disease of fundamental importance for the global community. The aim of these projects is to find a possible solution to restore the functionalities lost by a patient subjected to an upper limb amputation trying to develop, according to physiological considerations, a communication link between the brain in which the significant signals are generated and a motor prosthesis device able to perform the desired action. Moreover, the designed system must be able to give back to the brain a sensory feedback about the surrounding world in terms of pressure or temperature acquired by tactile biosensors placed at the surface of the cybernetic hand. It in fact allows to execute involuntary movements when for example the arm comes in contact with hot objects.

The development of such a closed-loop architecture involves the need to address some critical issues which depend on the chosen approach. Several solutions have been proposed by the researches of the field, each one differing with respect to where the neural signals are acquired, either at the central nervous system or at the peripheral one, most of them following the former even that the latter is always considered by the amputees a more natural way to handle the artificial limb. This research work is based on the use of intrafascicular electrodes directly implanted in the residual peripheral nerves of the stump which represents a good compromise choice in terms of invasiveness and selectivity extracting electroneurographic (ENG) signals from which it is possible to identify the significant activity of a quite limited number of neuronal cells. In the perspective of the hardware implementation of the resulting solution which can work autonomously without any intervention by the amputee in an adaptive way according to the current characteristics of the processed signal and by using batteries as power source allowing portability, it is necessary to fulfill the tight constraints imposed by the application under consideration involved in each of the various phases which compose the considered closed-loop system.

Regarding to the recording phase, the implementation must be able to remove the unwanted interferences mainly due to the electro-stimulations of the muscles placed near the electrodes featured by an order of magnitude much greater in comparison to that of the signals of interest amplifying the frequency components belonging to the significant bandwidth, and to convert them with a high resolution in order to obtain good performance at the next processing phases. To this aim, a recording module for peripheral neural signals will be presented, based on the use of a sigma-delta architecture which is composed by two main parts: an analog front-end stage for neural signal acquisition, pre-filtering and

sigma-delta modulation and a digital unit for sigma-delta decimation and system configuration. Hardware/software cosimulations exploiting the Xilinx System Generator tool in Matlab Simulink environment and then transistor-level simulations confirmed that the system is capable of recording neural signals in the order of magnitude of tens of μV rejecting the huge low-frequency noise due to electromyographic interferences.

The same architecture has been then exploited to implement a prototype of an 8-channel implantable electronic bi-directional interface between the peripheral nervous system and the neuro-controlled hand prosthesis. The solution includes a custom designed Integrated Circuit (0.35μm CMOS technology), responsible of the signal pre-filtering and sigma-delta modulation for each channel and the neural stimuli generation (in the opposite path) based on the directives sent by a digital control system mapped on a low-cost Xilinx FPGA Spartan-3E 1600 development board which also involves the multi-channel sigma-delta decimation with a high-order band-pass filter as first stage in order to totally remove the unwanted interferences. In this way, the analog chip can be implanted near the electrodes thanks to its limited size avoiding to add a huge noise to the weak neural signals due to long wires connections and to cause heat-related infections, shifting the complexity to the digital part which can be hosted on a separated device in the stump of the amputee without using complex laboratory instrumentations. The system has been successfully tested from the electrical point of view and with in-vivo experiments exposing good results in terms of output resolution and noise rejection even in case of critical conditions.

The various output channels at the Nyquist sampling frequency coming from the acquisition system must be processed in order to decode the intentions of movements of the amputee, applying the correspondent electro-mechanical stimulation in input to the cybernetic hand in order to perform the desired motor action. Different decoding approaches have been presented in the past, the majority of them were conceived starting from the relative implementation and performance evaluation of their off-line version. At the end of the research, it is necessary to develop these solutions on embedded systems performing an on-line processing of the peripheral neural signals. However, it is often possible only by using complex hardware platforms clocked at very high operating frequencies which are not be compliant with the low-power requirements needed to allow portability for the prosthetic device.

At present, in fact, the important aspect of the real-time implementation of sophisticated signal processing algorithms on embedded systems has been often overlooked, notwithstanding the impact that limited resources of the former may have on the efficiency/effectiveness of any given algorithm. In this research work it has been addressed the optimization of a state-of-the-art algorithm for PNS signals decoding that is a step forward for its real-time, full implementation onto a floating-point Digital Signal Processor (DSP). Beyond low-level optimizations, different solutions have been proposed at an high level in order to find the best trade-off in terms of effectiveness/efficiency. A latency model, obtained through cycle accurate profiling of the different code sections, has been drawn in order to perform a fair performance assessment. The proposed optimized real-time algorithm achieves up to 96% of correct classification on real PNS signals acquired through tf-LIFE electrodes on animals, and performs as the best off-line algorithm for spike clustering on a synthetic cortical dataset characterized by a reasonable dissimilarity between the spike morphologies of different neurons.

When the real-time requirements are joined to the fulfilment of area and power minimization for implantable/portable applications, such as for the target neuroprosthetic devices, only custom VLSI implementations can be adopted. In this case, every part of the algorithm should be carefully tuned. To this aim, the first preprocessing stage of the decoding algorithm based on the use of a Wavelet Denoising solution able to remove also the in-band noise sources has been deeply analysed in order to obtain an optimal hardware implementation. In particular, the usually overlooked part related to threshold estimation has been evaluated in terms of required hardware resources and functionality, exploiting the commercial Xilinx System Generator tool for the design of the architecture and the co-simulation. The analysis has revealed how the widely used Median Absolute Deviation (MAD) could lead to hardware implementations highly inefficient compared to other dispersion estimators demonstrating better scalability, relatively to the specific application.

Finally, two different hardware implementations of the reference decoding algorithm have been presented highlighting pros and cons of each one of them. Firstly, a novel approach based on high-level dataflow description and automatic hardware generation is presented and evaluated on the on-line template-matching spike sorting algorithm which represents the most complex processing stage. It starts from the identification of the single kernels with the greater computational complexity and using their dataflow description to generate the HDL implementation of a coarse-grained reconfigurable global kernel characterized by the minimum resources in order to reduce the area and the energy dissipation for the fulfilment of the low-power requirements imposed by the application. Results in the best case have revealed a 71% of area saving compared to more traditional solutions, without any accuracy penalty. With respect to single kernels execution, better latency performance are achievable still minimizing the number of adopted resources.

The performance in terms of latency can also be improved by tuning the implemented parallelism in the light of a defined number of channels and real-time constraints, by using more than one reconfigurable global kernel in order that they can be exploited to perform the same or different kernels at the same time in a parallel way, due to the fact that each one can execute the relative processing only in a sequential way. For this reason, a second FPGA-based prototype has been proposed based on the use of a Multi-Processor System-on-Chip (MPSoC) embedded architecture. This prototype is capable of respecting the real-time constraints posed by the application when clocked at less than 50 MHz, in comparison to 300 MHz of the previous DSP implementation. Considering that the application workload is extremely data dependent and unpredictable due to the sparsity of the neural signals, the architecture has to be dimensioned taking into account critical worst-case operating conditions in order to always ensure the correct functionality. To compensate the resulting over-provisioning of the system architecture, a software-controllable power management based on the use of clock gating techniques has been integrated in order to minimize the dynamic power consumption of the resulting solution.

Summarizing, this research work can be considered a sort of proof-of-concept for the proposed techniques considering all the design issues which characterize each stage of the closed-loop system in the perspective of a portable low-power real-time hardware implementation of the neuro-controlled prosthetic device.

Contents

Abstract **i**

Contents **v**

List of Figures **viii**

List of Tables **x**

1 Neuroprostheses Design: Problem Formulation **1**
1.1 Introduction 1
1.2 System Architecture 3

2 A Sigma-Delta Architecture for the Recording of Peripheral Neural Signals **7**
2.1 Introduction 7
2.2 Sigma-Delta A/D conversion 8
2.3 System Architecture 10
2.3.1 Design and modelling of the analog module 11
2.3.2 Design and realization of the digital module 13
2.4 Simulation Results 16

3 A Multi-Channel Electronic Interface for PNS Recording and Stimulation **21**
3.1 Introduction 21
3.2 System Architecture 22
3.2.1 Analog Front-End 22
3.2.2 Digital Back-end 24
3.3 Testing Environment Setup 26
3.3.1 Custom Designed Printed Circuit Board 28
3.3.2 Digital Control System 28
3.4 Experimental Results: electrical tests 35
3.4.1 Recording unit tests 35
3.4.2 Stimulation module tests 40
3.4.3 Real in-vivo tests 42

4 Real-time neural signals decoding onto off-the-shelf DSP processors **45**
4.1 Introduction 45
4.2 A state-of-art ENG-based signal processing algorithm 46

4.2.1 Wavelet Denoising 48
4.2.2 Spike Detection 52
4.2.3 Spike Sorting 53
4.2.4 Classification 55
4.3 The DSP implementation: porting details 57
4.4 Test Data 59
4.5 Experimental Results 60
4.5.1 Effectiveness analysis 61
4.5.2 Efficiency analysis 65

5 VLSI Wavelet Denoising of Neural Signals 71
5.1 Introduction 71
5.2 Algorithmic solutions for threshold estimation 72
5.3 Architecture Design exploiting Xilinx System Generator 73
5.4 Experimental Results 77
5.4.1 Functional Evaluation 78
5.4.2 Hardware Figures of Merit 80

6 A Coarse-Grained Reconfigurable Approach for Low-Power Neural Signal Decoding 85
6.1 Introduction 85
6.2 Exploiting the Multi-Dataflow Composer Tool 86
6.2.1 Multi-kernel datapath generation 86
6.2.2 HDL components library and Communication Protocol 88
6.2.3 Computing kernels 90
6.3 FPGA test environment 92
6.4 Experimental results 94
6.4.1 Accuracy results 94
6.4.2 Latency analysis 96
6.4.3 Area occupancy and power consumption 97

7 An FPGA-based MPSoC for On-line Neural Signal Decoding 101
7.1 Introduction 101
7.2 Related work 102
7.3 Target application and related constraints 103
7.3.1 Wavelet Denoising 105
7.3.2 Spike Detection 105
7.3.3 Spike Sorting 106
7.3.4 Real-time constraints 106
7.4 Parallelization and programming model 107
7.5 Sequential application profiling 108
7.6 Partitioning and mapping description 109
7.7 Hardware architecture 111
7.8 Integrated power consumption control 111
7.9 Experimental results 113
7.9.1 Timing Constraints Evaluation 113
7.9.2 Power consumption reduction 114
7.9.3 Synthesis Results 115

8 Conclusions **117**

Bibliography **121**

List of Figures

1.1 Neuroprosthetic solution: different approaches 2
1.2 Block diagram of the closed-loop system used for the neuroprosthesis control. . . 4
1.3 Typical electrodes able to acquire PNS neural signals. 5

2.1 Power Spectral Density of the Quantization Noise. 9
2.2 Typical block diagram of a Sigm-Delta Analog-to-Digital Converter. 9
2.3 Block diagram of a First Order Sigm-Delta Modulator. 10
2.4 System architecture 11
2.5 Behavioral Simulink model 12
2.6 Behavioural Simulink model with non-idealities and digital module 12
2.7 Blocks diagram of a 2^{nd} order CIC filter 14
2.8 Sigma-delta modulator: Power spectral density of the complete real model 17
2.9 Signal to Noise Ration variations with different signal amplitudes 17
2.10 Recording chain response of three input sines 19
2.11 Sigma-Delta modulator: pre-recorded neural signal processing 20

3.1 Block diagram of the neural recording and stimulation system 23
3.2 Eight-channels analog front-end mapped on the custom designed IC. 24
3.3 Block diagram of the multi-channel version of the digital BPF. 26
3.4 Architecture of the Digital Control System. 27
3.5 Implementation of the Testing Environment. 27
3.6 SPI communication protocol between the chip and the controller 30
3.7 Biphasic neural stimulus waveform 32
3.8 Set-up of the testing environment during stimulation. 33
3.9 Bode diagram of the reconfigurable analog band-pass prefilter 36
3.10 BPF: gain programmability. 36
3.11 Frequency response of the 16^{th} order digital BPF 37
3.12 Modulator and decimator output with a sinusoidal input with $0.5V$ at $5kHz$ . . . 38
3.13 SNR vs. Input Amplitude. 38
3.14 Recording system results in high-gain configuration 39
3.15 Pre-recorded neural signal processed by the recording module 40
3.16 Stimulation current amplitude programmability. 41
3.17 Stimulation biphasic pulses varying the relative phase width. 41
3.18 Stimulation biphasic pulses varying the relative period. 42
3.19 In-vivo tests at the EPFL laboratories. 42

3.20 In-vivo recording results with seven channels 43
3.21 In-vivo recording results: zoom on a single spike 43

4.1 Structure of the tf-LIFEs. 46
4.2 Schematic representation of the PNS decoding process on a real signal. 47
4.3 The proposed approach to the WD stage 51
4.4 The flow chart of the spike detection and the sorting steps. 54
4.5 SVM hyperplane which separates the features vectors belonging to the two classes. 56
4.6 Correlation among the various couples of spike templates for each dataset. 60
4.7 Spike detection results in terms of TP rate 62
4.8 Spike detection results in terms of FP per minute 63
4.9 Performance comparison of the various spike detection approaches 65
4.10 SVM classifier accuracy 66
4.11 The maximum of processable spikes per second in TPh 68
4.12 The maximum of processable spikes per second in SPh 69

5.1 Example of a system developed by Xilinx System Generator. 75
5.2 Simulink model of the wavelet denoising scheme using System Generator blocks. 75
5.3 Simulink Model of the Threshold Estimator block 76
5.4 Simulink Model of the sorter using an unfolded approach 76
5.5 Simulink Model of the sorter using an iterative approach 78
5.6 Simulink Model of the Threshold Estimator block 79
5.7 Wavelet Denoising input and outputs: low level noise, N=64 samples. 80
5.8 Wavelet Denoising input and outputs: high level noise, N=64 samples. 81
5.9 Wavelet Denoising input and outputs: low level noise, N=128 samples. 82
5.10 Threshold variation over time using different observation windows 83

6.1 Coarse-grained reconfigurable platform composition flow. 88
6.2 Floating Point Adder: the Block Diagram of the Two Path algorithm. 89
6.3 Multi-kernel datapath of the communication protocol. 90
6.4 The flow diagram of the implemented algorithm. 91
6.5 Block diagram of the proposed wavelet denoising solution 91
6.6 Single-stage decomposition dataflow kernel, *dec*. 91
6.7 Single-stage recomposition dataflow kernel, *rec*. 92
6.8 Thresholding dataflow kernel, *thr*. 92
6.9 Test system block diagram. 93
6.10 Input and output denoised signals. Low level noise. *Haar* mother wavelet 95
6.11 Input and output denoised signals. High level noise. *Haar* mother wavelet 96
6.12 Input and output denoised signals. Low level noise. *Debauchies 2* mother wavelet. 96
6.13 Latency comparison between the totally software and the coarse-grained solutions 97

7.1 Flow diagram of the target neural signal processing algorithm 104
7.2 The Wavelet Denoising scheme. 105
7.3 Application graph resulting from the application partitioning 110
7.4 Custom MPSoC architecture implemented on the FPGA device. 111
7.5 Power consumption of the system measured on the FPGA 113
7.6 Execution trace of the multi-core system. 114

List of Tables

2.1 Sigma-delta modulator: coefficients 13
2.2 Digital design parameters 16

3.1 Digital filter parameters 25
3.2 Synthesis results in case of 8 parallel channels and the multiplexed solution 26
3.3 Biphasic neural stimulus waveform 32
3.4 BPF parameters 36

4.1 Template matching percentage using the Haar and the Symlet 7 mother wavelet. . 64

5.1 FPGA synthesis results for the *Threshold Estimator* 83

6.1 Number of spikes for each template 97
6.2 Number of adder and multiplier FUs of the wavelet denoising implementations . 98
6.3 Synthesis results for the most interesting FUs. 99
6.4 FPGA synthesis results only considering the kernels for the wavelet denoising . . . 99
6.5 FPGA synthesis results considering all the kernels of the algorithm. 99
6.6 ASIC synthesis results. 100
6.7 ASIC synthesis results. 100

7.1 Execution time of the main processing steps. 109
7.2 Percentage of time with enabled Normalized cross-correlators 115
7.3 FPGA Synthesis 116

Chapter 1

Neuroprostheses Design: Problem Formulation

1.1 Introduction

Neuroprostheses aim at restoring the functionalities lost by a patient subjected to an upper limb amputation in order that he/she is able to perform the common actions which characterize the daily life of a person without this disease. It involves the possibility of grasping and manipulate objects or receiving sensorial feedback from the surrounding world in terms of temperature or pressure when the cybernetic hand comes into contact with these ones in comparison to other previous commercial solutions which make sense only from the cosmetic point of view. Most of the efforts made on this research field have been done thanks to the large investments funded by American institutions due to the huge number of mutilated veterans, especially after the conflict in Vietnam, subjected to these amputations. The interest has been then expanded across the entire international research world, getting a massive attention from the experts of the field.

Several approaches have been presented in the years obtaining even an academic and a commercial dissemination. However, none of these has determined the development of a versatile prosthetic limb which can be controlled in an intuitive way and which can be considered by the patients as a real part of their body, without having critical functional conditions due to, for example, phantom limb syndromes. The objective is to create a bidirectional communication link between the brain and the artificial peripheral cybernetic hand analysing the information encoded into physiological measurements, even in case of low signal-to-noise ratio conditions and generating the correspondent electro-mechanical stimulations which must be given in input to the prosthesis.

The neural activity can be captured at the level of either the Central Nervous System (CNS) or the Peripheral Nervous System (PNS) [82]. The decoded intention is usually used to operate mechatronic prostheses: exploiting the natural pathways of motor control, they are more easily and finely managed (and then accepted) by the amputees compared to traditional Electromyographic (EMG) controlled ones which instead exploit the stimulations at the surface of the muscles, normally not involved in the common process of handling the upper limb. This is due to the fact that the human brain does not adapt easily to the myoelectric control of the upper limb prosthesis because the muscles, on whose surface the signals of interest are extracted (for instance, bicep or tricep muscles), are not normally exploited to direct the movements of the hand. However, EMG-based prosthesis represent the solution which have had the most academic and commercial dissemination [3, 2] thanks to its limited invasivity even if this method allows only a single motion at a time needing that the operations must be done sequentially.

Beyond approaches based on the processing of the Electroencephalographic (EEG) signal [62] through the development of customized Brain-Machine Interfaces (BMIs), all the others require invasive procedures to access the neural signals. In this case, the decoding techniques are strongly influenced by the chosen neural interface and the consequent selectivity [88, 74, 16]. Selectivity, in this context, is inversely proportional to the number of motor neurons whose firing activity (i.e. the development of action potentials, usually called *spikes*) is captured on a single channel of the neural interface. The higher the selectivity, the higher the possibility of identifying the activity of the single neurons through *spike sorting* techniques looking at the action potential morphology as the fingerprint of different neurons [46], and then the finer will be the control of the prosthesis.

(a) EEG-based Brain Machine Interfaces

(b) EMG-based solution

(c) Targeted Muscle Reinnervation

(d) ENG-based solution

Figure 1.1: Neuroprosthetic solution: different approaches

For these reasons, prostheses controlled by Electroneurographic (ENG) signal at PNS level is surely the most promising approach thanks to the fact that they must access directly to the peripheral residual nerves of the stump to acquire the significant signal extracted by the implanted electrodes [60]. It helps the amputees to consider the available cybernetic hand as an effective part of their body, avoiding situations of rejection crisis. Moreover, this last approach has the advantage that the ENG signal provides much more information in comparison to the other signal sources allowing to control more degrees of freedom but at the cost of greatly complicating the necessary neural signal processing algorithms.

EMG-based prostheses exploiting Target Muscle Reinnervation (TMR) techniques [40, 39] represent a compromise design choice between the two considered solutions trying to take advantage of the benefits provided by them. TMR uses the peripheral residual nerves of the amputated upper limb and redirects them onto other muscle groups that are not functional due that they are no longer to the missing arm. In particular, the median nerve can be retargeted on the pectoral muscle decoding the information related to the desired motor action from the signals coming from a grid of monopolar surface electrodes implanted on the reinnervated muscle by a difficult surgery procedure. With a proper training, the amputee is able in a few days to use in the correct way the prosthetic hand, closing or opening it by only contracting the pectoral muscle. Moreover, a bidirectional interface can be obtained by using in the same way the Targeted Sensory Reinnervation (TSR). It adopts sensors in the prosthesis which can quantify temperature or pressure and transmit these measurements over the reinnervated skin so that the amputee can have a sense of touch in the missing limb. As said before, in this case the patient must give his/her written informed consent to the difficult surgery procedure due to its possible risks which can cause for example the permanent paralysis of the target muscle and the recurrence of phantom limb pain.

Taking into account these considerations, in the following it will be considered the ENG-controlled upper limb prosthesis as the reference solution trying to address in an effective way the issues which characterize the various processing stages involved in the proposed solution to allow the realization of the relative hardware implementation.

1.2 System Architecture

The design of a motor neuroprosthesis involves the implementation of a close-loop system which can work autonomously without any external setting performed by the amputee. Fig. 1.2 shows the block diagram of its typical architecture in which it is possible to highlight the presence of both an efferent path from the CNS to the PNS and, in the opposite direction, of an afferent one from the PNS to the CNS. It determines the creation of a bidirectional communication link between the brain in which the neural signals are generated and the cybernetic hand which represents the actuator of the movements required by the patient.

The aim of the direct path is to acquire the neural signals coming from the brain, convert them from the analog to the digital domain with an acceptable resolution even in case of low signal-to-noise ratio conditions and decoding the intentions of movement by analysing the information encoded into the physiological measurements, for example in terms of spikeness of the significant neuronal cells in order to properly apply the correct electro-mechanical stimulations in input to the prosthesis device. At the same time, the opposite path must be able to give to the amputee a sensorial feedback about the outside world in terms of, for example, pressure or temperature values coming from tactile biosensors placed in the external surface of the cybernetic hand. This research work is focused on the design and the development of the compact portable low-power electronic devices, especially with a greater attention for those belonging to the digital domain, which must be involved in this closed-loop system.

As can be seen in Fig.1.2, the general processing flow adopted for the implementation of a neuroprosthesis at PNS level is characterized by the following phases:

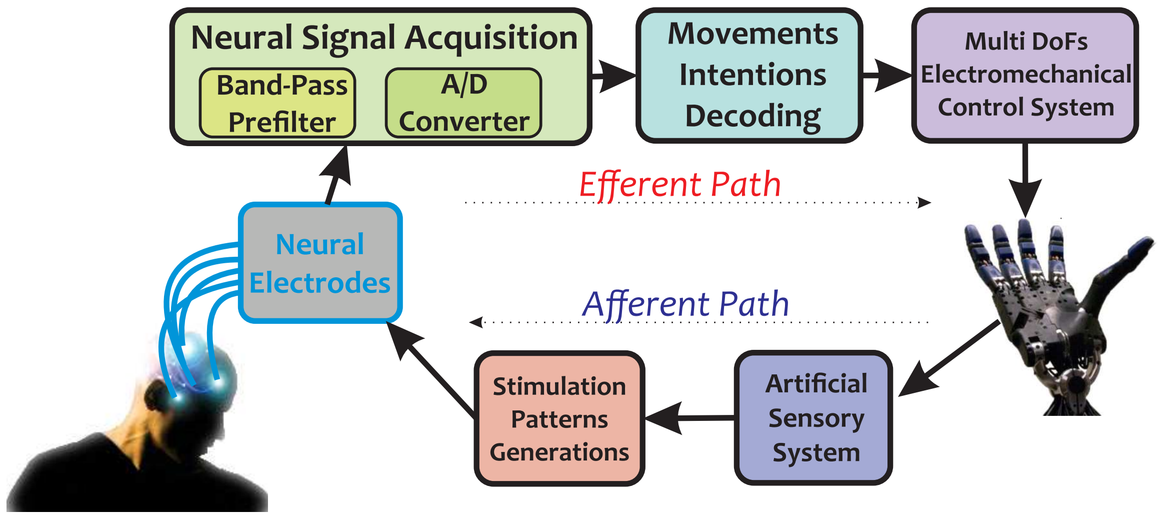
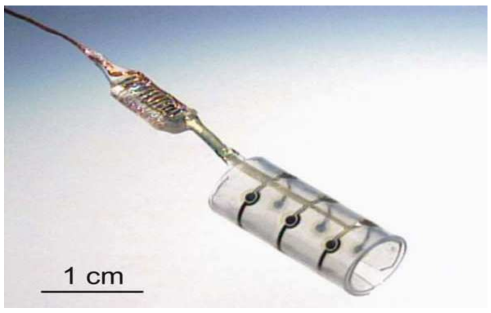

Neural Signal Acquisition
Band-Pass Prefilter
A/D Converter
Movements Intentions Decoding
Multi DoFs Electromechanical Control System
Efferent Path
Neural Electrodes
Afferent Path
Stimulation Patterns Generations
Artificial Sensory System

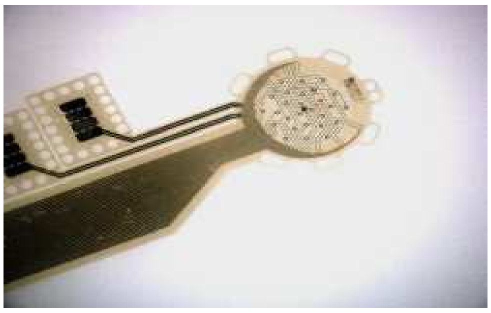
Figure 1.2: Block diagram of the closed-loop system used for the neuroprosthesis control.

- the *acquisition* of the neural signals through the direct implantation of multi-channel electrodes on the peripheral residual nerves accessible from the stump of the amputee;
- the *recording* of the neural signals which includes a low-order pre-filtering stage to partially remove the unwanted interferences and the analog-to-digital conversion with high resolution trying to relax the timing constraints at downstream;
- the *pre-processing* phase for the removal of the in-band and the out-of-band noise overimposed on the significant signal and the *decoding* of the intentions of movement of the amputee through the implementation of complex digital signal processing algorithms;
- the generation of the corresponding electro-mechanical stimulations to give in input to the cybernetic hand;
- finally in the opposite path, starting from the external sensory information coming from the tactile biosensors, determines the generation of stimulation patterns in the form of bi-phasic pulses with variable morphology and frequency which must be given in input to the same electrodes used for the recording phase.

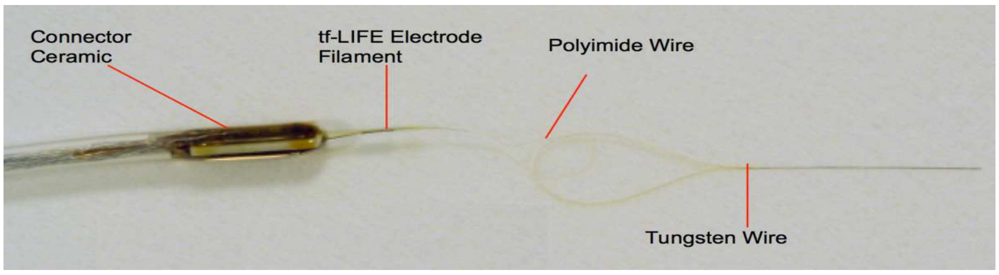
Surely, the choice of the electrodes used to acquire the neural signals coming to the peripheral nerves strongly influences the design of the following recording and decoding stages because each one of them has particular characteristics in terms of selectivity and output significant bandwidth. For example, exploiting multi-site Cuff electrodes as in [49] it is not possible to determine the activity of each neuronal cell from the extracted physiological measurements even if they have a low level of invasiveness. This is due to the fact that these ones represent a sort of cumulative signals of the overall activity generated by the neuronal cells from which the intentions of movements of the amputee can be decoded only by applying adaptive amplitude thresholding or adopting complex unsupervised clustering algorithms without achieving several degrees of freedom in terms of possible grasps of the cybernetic hand. On the contrary, Sieve electrodes [42] are characterized by high performance in terms of selectivity but, at the same time, several unresolved problems with their chronic stability and the requirement to sectioned nerves which limit their usability in real applications which can degrade the decoding performance. Therefore, an acceptable compromise design solution can be instead the use of Longitudinal Intra-Fascicular electrodes (tf-LIFEs) [58] for their level of selectivity and invasiveness.



(a) Cuff Electrodes



(b) Sieve Electrodes



(c) tf-LIFE Electrodes

Figure 1.3: Typical electrodes able to acquire PNS neural signals.

Good performance of the recording stage are also necessary for the correct implementation of the prosthesis device because it must be able to reject the huge interferences overlapped to the signals of interest which have typically an order of amplitude significantly lower in comparison to the unwanted sources. For this reason, a pre-filtering stage with a programmable gain is usually mandatory. Regarding to the decoding step, most of the previous works were based on the use of spike sorting algorithms in order to obtain the intentions of movements of the amputees according to the significant activity of each single neuron. These algorithms were conceived starting from the development of the relative off-line implementation determining a significant drop of the achievable performance when they must be modified in order to allow an on-line processing. Other solutions were not absolutely portable in a hardware implementation that can work on-line so the results presented do not have a practical sense since the purpose of the research is not to use solutions that will work well only on work stations. It is due to the need of the fulfilment of real-time constraints especially when they are characterized by techniques with high computational complexity. These aspects have been often overlooked in the literature and will be deeply considered in this thesis.

This research work has in fact focused its efforts in the study, the design and the development of hardware implementations related to the neural signal acquisition aimed at the recording and stimulation phases and the decoding of peripheral neural signals for the physical realization of the prosthetic device. The target is to realize hardware solutions which can be partially implanted near the electrodes, avoiding heat-related infections and the overimposing of relevant noise sources to the signals of interest. The remaining part can be mapped on low-power and low-area embedded systems able to perform complex digital signal processing algorithms at a reasonable operating frequency allowing portability using batteries as power source for the resultant implementation. The thematics related to the power consumption has been in fact exacerbated, in particular when it is necessary to place in the stump of the amputee both the acquisition system and the processing element. The development of implantable solutions is motivated by the need of avoiding long wired connections between the neural implant and the external world. A wireless link can not be always possible due to the huge amount of data which must be transmitted through a link with a limited useful bandwidth.

In this way, it will be possible to satisfy the tight real-time constraints imposed by the reference application even exploiting hardware platforms with embedded electronics characterized by limited resources for recording, processing and stimulation in comparison to other previous solutions which however needed the use of powerful laboratory instruments to perform the various tasks involved in the closed-loop system. This work would be a sort of proof-of-concept for the proposed approaches in order to finally implement a versatile solution which can have in the future perspective a wide dissemination from the academic and the commercial point of view thanks to its limited cost.

The remainder of this thesis is as the following. In the Chap.2 will be proposed a sigma-delta analog-to-digital conversion architecture able to properly record the weak signals of interest, evaluating the relative performance in terms of output resolution and noise rejection by accurate hardware/software cosimulations and then by transistor-level simulations. The same architecture will be exploited in the Chap.3 to implement a multi-channel bidirectional bio-electronic interface to acquire and digitalize the physiological signals coming from the electrodes and to generate stimulation patterns of bi-phasic pulses restoring the sensory feedback to the amputee, involving the use of a custom designed Integrated Circuit for the analog part and an FPGA hosting the digital part and a control system able to handle the tests from the electrical point of view and the various phases during in-vivo experiments. In the Chap.4, a state-of-the-art decoding algorithm will be ported into a floating-point off-the-shelf DSP to verify the fulfilment of the tight real-time constraints imposed by the application. With the aim of a perspective ASIC low-power implementation, its first processing step based on the use of a Wavelet Denoising algorithm will be tuned in the Chap.5 in order to obtain the optimal solution for the costly-operation thresholding estimation. Consequently, two alternative hardware implementations will be presented to satisfy the low-power requirements for the realization of a portable solution able to work autonomously with batteries as power source and without using high operating frequency, respectively the first one based on a coarse-grained reconfigurable architecture in the Chap.6 and the second one adopting a Multi-Processor System-on-Chip with an efficient power consumption control by the use of a software-controlled clock-gating manager. Finally, in the Chap.8 a final discussion will be reported summarizing the achieved results with respect to the required performance.

Chapter 2

A Sigma-Delta Architecture for the Recording of Peripheral Neural Signals

2.1 Introduction

To allow the multi-channel acquisition of the peripheral neural signals coming from the electrodes implanted in the residual nerve of the stump, it is necessary to develop a dedicated hardware system with great performance in terms of resolution and noise rejection even in case of low signal-to-noise ratio (SNR) conditions. At the same time, it is necessary to fulfil strict constraints in terms of area and power constraints imposed by the fact that the recording module must be placed near the electrodes avoiding of causing infections due to overheating of the skin tissues.

It must be able to acquire the peripheral neural signals with amplitudes in the order of tens of μV, subjected to noise due to the contractions of the muscles near the electrodes (EMG interferences). Such noise is in the order of magnitude of mV and can thus mask the neural signal. Furthermore, Power Spectral Densities (PSDs) of the interferences and of the useful signal are very close to each other and partially overlap [75, 26]. To this aim, it is necessary to design a high-pass frequency response with a sharp transition band in order to reject these interferences which can limit the performance of the next processing steps in the digital domain but these requirements do not allow implantable or implantable solutions. Therefore, the recording system must be able in real-time to acquire the signals at output of the electrodes, to digitalize them with the best allowable resolution rejecting the interferences which are not useful and finally to transmit the relative samples in input to the device responsible of the decoding of the movements intentions performed by optimal complex digital signal processing algorithms mapped on embedded systems.

An analog front-end amplification and filtering stage is then mandatory in order to boost the weak neural signal and to filter out the huge EMG components. Many papers concerning neural interfaces have been presented in literature, but the majority of them are focused on the CNS [27, 34]. However, in this last case, the relative solutions are not influenced by these physiological interferences but only the background activity of the neurons at a greater distance from the acquisition electrodes due to their limited selectivity. Regarding to the PNS recording, the most used approach is oriented to a a fully analogical implementation based on multi-stage high selective filters [47, 45, 86] followed by standard, Nyquist-rate, Analog to Digital Converters (ADC) while an alternative approach is based on oversampling converters using a sigma-delta architecture. For example, in [90] a first order sigma-delta converter has been presented, reaching an 8-bit resolution with a 40 oversampling ratio over a $6.25kHz$ frequency whereas, in [85], a second order sigma-delta modulator that exploits a new super-inverter amplifier that allows to reach a 11 bit resolution considering a $8kHz$ bandwidth.

Following the latter approach, it has been implemented a recording system based on a sigma-delta architecture which combines high resolution and high integration capacity altogether with the possibility of easily decoupling the sensitive and potentially implantable analog module and the robust external digital module which can placed in a different device. Sigma-delta oversampling converters are particularly suited for low bandwidth applications since they are capable of increasing the achieved resolution by increasing the sampling frequencies, as can be seen below. Moreover, they shift the design complexity from the analog to the digital domain exploiting the greater integration capability of this domain thanks to the improvements in terms of CMOS processes performed in these last years. In this way, it is possible to integrate on an implantable chip only a limited number of simple, low-power, low-noise analog components near the electrodes. Such components include a pre-filtering stage and the sigma-delta modulator, made-up of integrators and a 1-bit quantizer. The output signal is represented by the 1-bit output of the quantizer for each channel thus requirements for the communication channel are less stringent.

The complex digital unit that finalizes the A/D conversion and provides the highly selective band-pass filter needed to remove the EMG interferences at low frequencies, can thus be accommodated on an external digital module such as a Field-Programmable Gate Array (FPGA) or a low-power custom designed Application Specific Integrated Circuit (ASIC). In the following, it will be reported a short introduction to the sigma-delta conversion, the description of the adopted system architecture tested through hardware-software cosimulations exploiting the Xilinx System Generator tool on the user-friendly Matlab Simulink environment for the digital part taking into consideration an accurate software model for the analog part and, finally, some experimental results will be properly discussed.

2.2 Sigma-Delta A/D conversion

Sigma-delta converters offer high resolution and high integration, making them a good design choice for a wide range of applications. They are based on an architecture which allows to obtain low cost conversion with high dynamic range and flexibility especially in converting low bandwidth input signals such as in the considered target application for the recording of the ENG-based peripheral neural signals.

This solution tries to improve the achieved resolution and the SNR value at output of the Analog-to-Digital converter in comparison to other traditional implementations at the cost of complicating the relative digital part, exploiting two main theoretical concepts, *oversampling* and *noise shaping*. Taking into account an approximated model, a quantizer with N output bits can be considered as a linear system which introduces the addictive quantization noise with a constant-distributed Power Spectral Density (PSD) in the frequency domain (white noise) assuming that it has a random behaviour which overlaps the useful bandwidth of the signal of interest. If the input signal which must be converted has a limited bandwidth until the frequency f_0, the Nyquist theorem imposes the minimum sampling rate f_s at $2f_0$ while in the case of a sigma-delta converter, it must be fixed to a larger value spreading the quantization noise over a wider spectrum. The latter is still the same but the relative percentage in the band of interest of the input signal is greatly reduced. Fig.2.1 shows the PSD of the quantization noise and the SNR of an A/D converter in the case of respectively a Nyquist and an oversampling solution where OSR indicates the oversampling factor $(f_s/2f_0)$, equal to 1 in the first case.

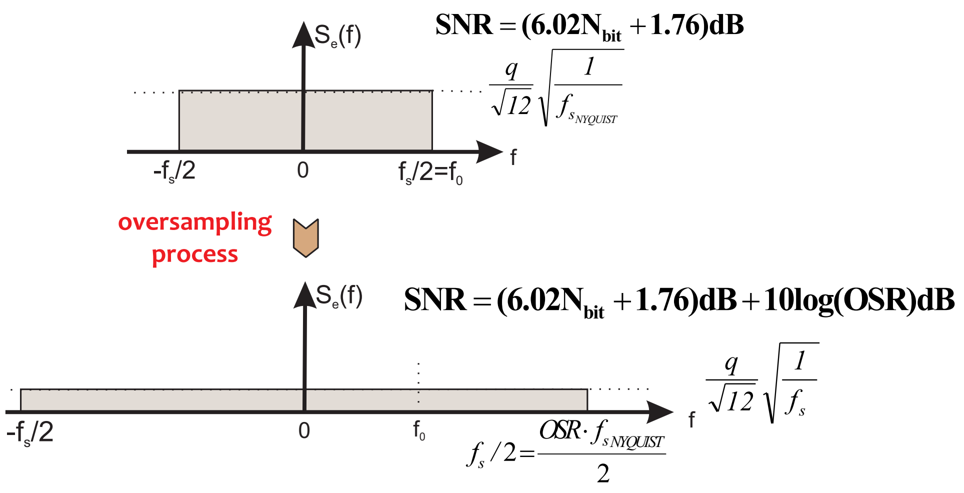


Figure 2.1: Power Spectral Density of the Quantization Noise.

Regarding to the noise shaping, it is based on the fact that the signal and the quantization noise are filtered by the system in two different ways. In particular, the former is applied to a low-pass filtering process whereas the latter is applied to a high-pass one in order that the latter is shifted at the frequencies above the significant bandwidth of the considered signal and which can be removed by the next processing stages in the digital domain thanks to its greater integration capacity exploitable to obtain high-selective filters. Fig.2.2 shows the typical block diagram of a sigma-delta A/D converter which is composed by an analog front-end and a digital back-end: the analog part involves a low-pass filter to prevent aliasing effects due to the next sampling process at the oversampling frequency f_s and a sigma-delta modulator generating at output a 1-bit stream whereas the digital one involves a sigma-delta decimator which brings back the sampling frequency at the Nyquist frequency.

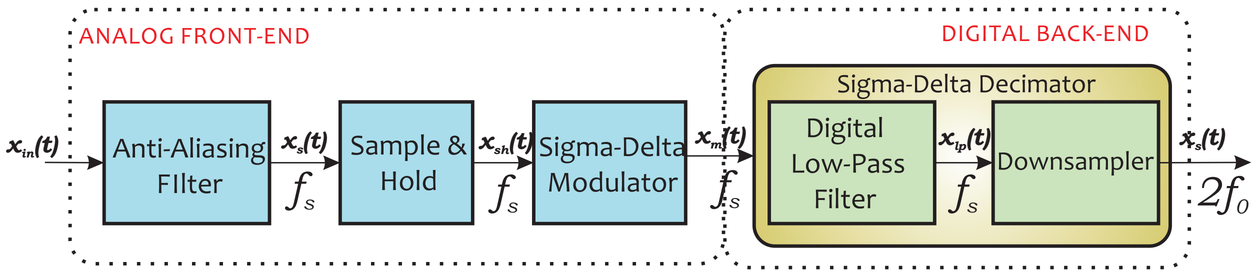


Figure 2.2: Typical block diagram of a Sigm-Delta Analog-to-Digital Converter.

Taking into account the architecture of a sigma-delta modulator, as can be seen in fig. 2.3 for a first order solution, it is possible to demonstrate that it performs a low-pass filtering on the input signal while a high-pass one on the quantization noise, shifting the significant frequency components above the band of interest of the considered signal. It is the aim of the digital part to totally remove the out-of-band contribution of the quantization noise through a high-order low-pass filter which must avoid aliasing effects before the downsampling process and remove the quantization noise shifted at high frequencies by the noise shaping. In the particular case, the first stage of the sigma-delta decimator must involve even the use of a high-order high-pass filter in order to totally remove the significant electro-stimulations of the muscles placed near the electrodes (EMG interferences) and other low frequency interferences.

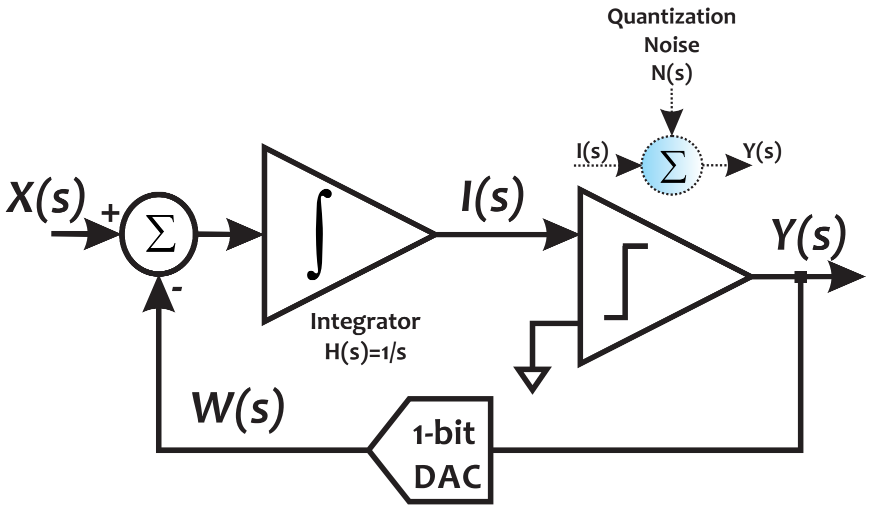


Figure 2.3: Block diagram of a First Order Sigm-Delta Modulator.

Due to the fact that the resultant digital band-pass filter works at the oversampling frequency, the relative dynamic power consumption is very high at the increase of the order and the number of the processing channels which depends on the type of electrodes used to extract the physiological signals. For this reason, over the years, alternative decimation filters architecture have been proposed by the researchers of the field trying to limit the relative number of hardware resources in order to map them in a low-power embedded platform and hosted in the stump of the amputee.

2.3 System Architecture

The block diagram in Fig. 2.4 shows the multi-channel system architecture of the proposed neural recording module, following the scheme specified in the previous section, which is composed of two main components: the analog front-end and the digital back-end.

The multi-channel signal comes from the various sensitive sites of the electrodes which must be implanted directly in the residual nerve of the stump in order to acquire the relevant activity of the neurons. In this proposed block diagram it has been considered the use of thin-film Longitudinal Intra-Fascicular Electrodes (tf-LIFEs) which have presented in previous works as in [59] very good performance in terms of selectivity determining relevant results in terms of decoding of intentions of movements for the next processing stages. This signal is, thus, filtered and amplified by a $1st$ order Band-Pass Pre-Filtering/Pre-amplifier

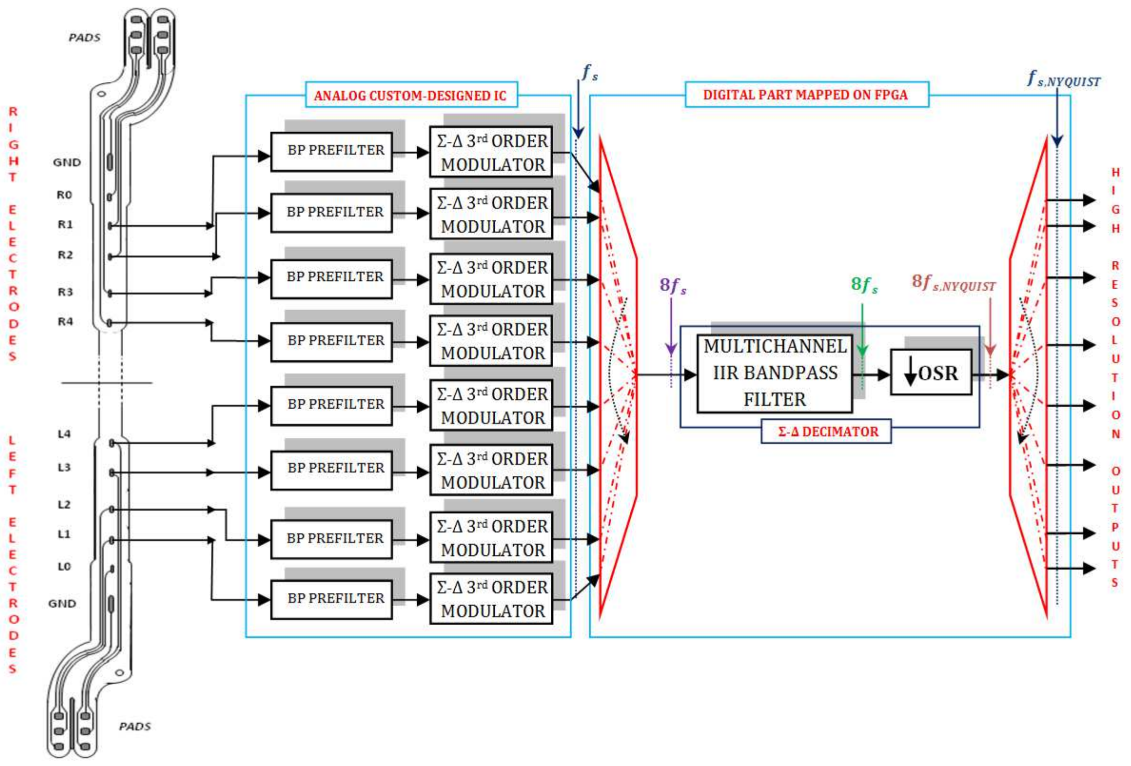


Figure 2.4: System architecture

block as close as possible to the recording site. In this way, the noise due to cables and connection paths can be avoided. The conditioned signal is then converted into a 1-bit digital stream for each channel by the sigma-delta modulator and sent to the digital module for decimation and further processing. One of such signal's streams is needed for each input channel if the device is connected to a multichannel electrode. The digital module is hosted on an external board; it implements the decimation block of the sigma-delta converter altogether with the highly selective band-pass filter. The digital module is also responsible of the management of the communication between the artificial limb and the implanted electrodes. The hardware digital unit has been implemented and tested on a Xilinx FPGA Virtex-5 LX330 to be hosted on the robotic limb thanks to its considerable amount of available resources.

2.3.1 Design and modelling of the analog module

As said before, the analog part of the circuit is composed by the pre-amplifier/pre-filtering block and by a third order sigma-delta modulator (Fig. 2.5). The 1^{st} order Band-Pass Filter (BPF) was realized cascading a High-Pass Filter (HPF) with a Low-Pass Filter (LPF). The frequency response specifications require a bandwidth between $800Hz$ and $8kHz$ which is considered the useful bandwidth for the peripheral neural signals of interest and a gain of $200V/V$. The required gain is determined by the need to maximize the amplification at neural signal frequencies avoiding the risk of amplifier saturation due to EMG interferences. Once that the useful signal has been properly amplified it is possible to convert it in the digital domain. With this purpose, a 3^{rd} order switched-capacitor, sigma-delta modulator in a

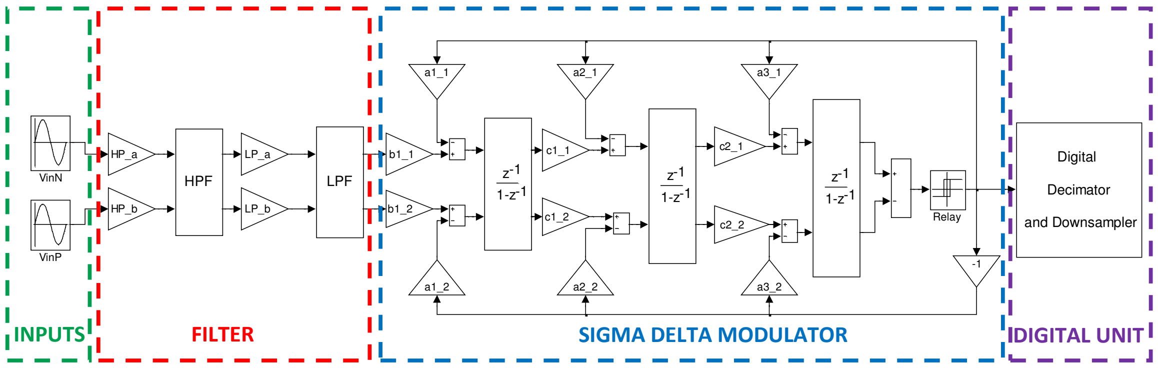


Figure 2.5: Behavioral Simulink model

Cascade of Integrators with FeedBack (CIFB) configuration was designed. Preliminary tests were performed using a behavioural model (shown in Fig. 2.5) in Simulink environment and, only once that the specifications were satisfied, the circuit was implemented at transistor level in a 0.35μm CMOS technology process from Austriamicrosystems (AMS).

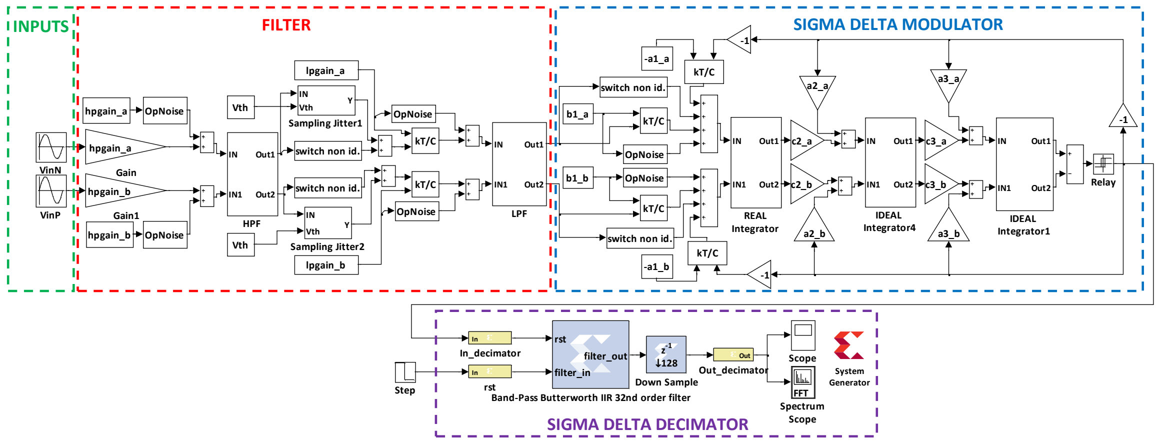


Figure 2.6: Behavioural Simulink model with non-idealities and digital module

The Simulink model has great advantages in terms of simulation time saving and it allows the modelling of noise sources and operational amplifier (OpAmp) non-idealities, thus ensuring a good agreement with transistor level simulations exploiting its user-friendly environment which allows to design the system at a higher level. Moreover, it allows to perform hardware/software cosimulations exploiting the Xilinx System Generator tool to design the digital part which can be mapped on the target FPGA device, handling the communication with the software blocks used for the analog part and having a high-level model view before the realization of the successive totally hardware implementation.

The sigma-delta behavioural model is a modification of what presented in [52] and [91] which take into consideration saturation, slew rate, finite gain and bandwidth limitations. KT/C noise, thermal noise of the amplifier, switches non-idealities and clock jitter effects have also been included. The original models were adapted to accurately model the switched-capacitors, fully-differential architecture of the implemented circuit. Component's mismatch effects were also modelled by generating the filter coefficients as capacitor ratios whose values have been extracted randomly from a normal distribution within a 6σ range around the nominal value. Non-idealities of the switches take into account the use of transmission gates

Table 2.1: Sigma-delta modulator: coefficients

Coefficient	Value	Coefficient	Value
a_1	0.05	b_1	0.05
a_2	0.3	c_1	1
a_3	0.8	c_2	1

(pair of NMOS-PMOS switches) and the clock jitter has been modified in order to consider the differential path of the fully-differential architecture. The complete model, including all non-idealities and the digital decimation described later is shown in Fig. 2.6.

The HPF and LPF transfer functions in the frequency domain (Eq. 2.1 and Eq. 2.2) can be easily determined once that cut-off frequencies have been defined ($\tau_{hp} = 1/f_{hp}$ and $\tau_{lp} = 1/f_{lp}$).

$$TF_{hp}(s) = \frac{\tau_{hp}s}{\tau_{hp}s+1} \quad (2.1) \qquad TF_{lp}(s) = \frac{1}{\tau_{lp}s+1} \quad (2.2)$$

The equivalent expression for the discrete time domain can be obtained using the bilinear transform (or Tustin's Method) based on the equivalence of Eq. 2.3

$$s \approx \frac{2}{T} \times \frac{z-1}{z+1} \tag{2.3}$$

The resulting transfer function are reported in Eq. 2.4 and Eq. 2.5

$$TF_{hp}(z) = \frac{2\tau_{hp} - 2\tau_{hp}z^{-1}}{(2\tau_{hp}+T) - (2\tau_{hp}-T)z^{-1}} \tag{2.4}$$

$$TF_{lp}(z) = \frac{T + Tz^{-1}}{(2\tau_{lp}+T) - (2\tau_{lp}-T)z^{-1}} \tag{2.5}$$

where T is the sample period and τ is the filter time constant.

The closed-loop transfer function is then expressed by Eq. 2.6, where $TF(z)$ is the filter transfer function of Eq. 2.1 or Eq. 2.2 and A is the finite gain of the open loop amplifier.

$$TF_{CL}(z) = \frac{TF(z)}{1 + \frac{TF(z)}{A}} \tag{2.6}$$

The coefficients (summarized in in Table 2.1) of the sigma-delta modulator were chosen using the Schreier Toolbox [77] with a 18-bits target resolution. The oversampling ratio that allows to reach this target resolution is $OSR = 128$ that, considering a signal bandwidth of $8kHZ$ results in a sampling frequency $f_s = 2.048MHz$.

2.3.2 Design and realization of the digital module

The main task of the digital block is to process the oversampled 1-bit signal provided by the analog modulator. In order to remove the EMG noise at low frequencies and the quantization noise pushed at high frequencies by the analog modulator, it is necessary to use a high-order band-pass anti-aliasing filter as the first decimator stage, before the downsampling process. The filter works at the same sampling frequency of the modulator which is equal to 2.048MHz, with a 128 oversampling ratio with respect to the Nyquist frequency (16kHz in our case). For these reasons, it is necessary to adopt some particular techniques in order to limit the area occupation and the dynamic power consumption in order to fulfill the tight constraints of the application under consideration in the perspective of the development of an ASIC solution which can work with batteries as power sources.

A classical approach optimized in terms of area occupancy and power consumption for the design of high-selective digital anti-aliasing filters in the case of low-pass sigma-delta A/D converters is based on a multi-stage structure using a Cascaded Integrator-Comb (CIC) filter [30]. This solution is widely used in commercial applications, especially in systems with multi-rate processing domains synchronized with respect to multiples of the same clock frequency.

In this case, the 1-bit stream at output of the sigma-delta modulator is given in input to the digital decimation filter that averages and downsamples it, thus producing an n-bit sample at the desired sampling rate. This process of averaging has the effect of low-pass filtering applied to the 1-bit signal in the frequency domain, which attenuates the quantization noise and removes aliases from the band of interest. It represents a Finite Impulse Response (FIR) filter and, at the same time, a cost-effective way of implementing decimation because it doesn't require multipliers but only addictions, subtractions and delay registers. Fig.2.7 shows the typical block diagram for a CIC filter in which R, M and N represent respectively the decimation factor (equal to the oversampling ratio), the differential delay and the number of sections of the digital filter. The transfer function of a common CIC filer is in the form of the Eq.2.7:

$$H(z) = \frac{(1 - z^{-RM})}{(1 - z^{-1})} \tag{2.7}$$

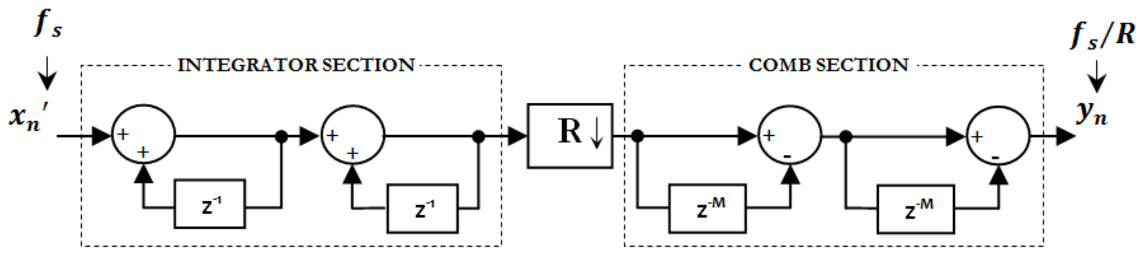


Figure 2.7: Blocks diagram of a 2^{nd} order CIC filter

As can be seen, the CIC architecture is very simple and it avoids the use of multipliers that are typically the greatest power consumers among the possible hardware digital modules. It includes the function of low-pass anti-aliasing filtering and downsampling at the same time. The number of sections N, that in the proposed example is equal to 2, influences the frequency response of the CIC filter. The higher the filter order, the more selective is the frequency response and consequently the greater the attenuation between the gain at low frequencies compared to the relative value at the first peak in stop-band frequencies. The position of the downsampler before the *Comb* section allows the use of a limited number of delay registers in this last section and a reduction of the dynamic power consumption thanks to the fact that these registers work at a lower clock frequency. In general, the typical value assigned for the differential delay parameter M is 1 or 2 and it effectively sets the number of nulls in the frequency response of a decimation filter.

It has been demonstrated in [30] that the CIC order must be chosen at least larger by one with respect to the order of the sigma-delta modulator. Moreover, to avoid overflow conditions it is necessary to use a 2's complement internal representation and the integrator word width has to be so long in order to handle the growing of data stored in the various registers of the CIC filter due to the relative DC gain higher than 1. Due to the fact that their frequency-magnitude-response envelopes are sin(x)/x-like, CIC filters are typically followed by higher performance linear-phase low-pass tapped-delay-line FIR filters whose tasks are to compensate the CIC filter's non-flat passband. In such a way, it is possible to obtain a total gain equal to 1 in the passband which could be a design constraint for real use cases.

However, in the target application, a band-pass solution is needed which can not be obtained with the use of only CIC filters. For these reasons, it has been decided to adopt a Butterworth Infinite Impulse Response (IIR) filter with a $3dB$ attenuation at $800Hz$ and $8kHz$ cut-off frequencies, as first stage of the sigma-delta decimator which allows to obtain good filtering techniques. IIR filters are useful for their high efficiency compared to that offered by FIR filters, in fact, they require less memory and fewer multipliers-accumulators. The main drawbacks are due to the instability problems, but they occur less likely using, as in this case, high order filters designed by cascading second-order sub-stages [38]. Moreover, in the perspective of a multi-channel processing context, if a Time Division Multiplexing (TDM) technique is used to reduce the number of necessary hardware resources for the resulting CIC filter, the gain in terms of slices saving becomes lower than what can be obtained for a single-channel implementation. In fact, for the CIC filters it is possible to share among the various channels only the logic cells related to adders and subtractors while the larger registers due to the not-null DC component must be replicated for each of them. In the IIR case, even those related to the multipliers can be shared.

Exploiting the Matlab tools, a 32^{nd} order Butterworth IIR band-pass filter has been created with the required frequency response and simulated in Simulink environment in order to evaluate if the output signal, pre-recorded during previous experimental tests, is correctly cleaned from the unwanted EMG interferences. A downsampler has been cascaded to the IIR filter; it is characterized by a simple Hardware Description Language (HDL) implementation, corresponding to a hardware module that picks up one sample every R which represents the oversampling factor equal to 128 in the considered case to bring back the sampling frequency at the Nyquist frequency $16kHz$. After verifying the correct functionality of the IIR filter on Simulink using a double-precision floating-point representation of the internal signals, the same behaviour has been verified in the case of fixed-point implementation. Values reported in Table 2.2 represent a good design parameter choice.

The decimation filter and the downsampler hav been properly coded in HDL and mapped on a FPGA in order to evaluate the percentage of the necessary resources and the behaviour of the whole system taking into account the hardware implementation of the digital part and the Simulink modules related to the analog one. A hardware-software co-simulation performed exploiting Xilinx System Generator (Fig. 2.6) has been set-up. In particular, the co-simulation allows to simulate the Simulink model of the analog module (including all non-idealities) and to send the results directly to the HDL implementation of the designed digital decimation filter mapped on the target FPGA. In this way, the software blocks gener-

Table 2.2: Digital design parameters

Response Type	Band-Pass Filter
Design Method	IIR Butterworth
Filter Arithmetic	Fixed-Point
Passband	0.8 - 8 KHz
Input Sampling Frequency	2.048 MHz
Order	32
Coefficient Word Length	32 bit
Input Word Length	32 bit
Input Fraction Length	20 bit
Output Word Length	32 bit
Rounding Mode	Floor
Overflow Mode	Wrap

ate the oversampled 1-bit stream that is set as input of a specific pin of the FPGA.

The tool allows to choose whether to use the hardware blocks coded by the user providing the relative HDL file or those provided by Xilinx. The latter only require to define the internal signal representation (fixed-point, unsigned or signed as 2's complement, etc.) as for the desired functionality, whereas the former must adhere to a standard interface mainly requiring an enable signal for each input clock to synchronize the modules inside the model due that the hardware blocks work at the clock frequency of the board whereas the software ones according to the Simulink simulation period. The Gateway In and the Gateway Out blocks delimit the hardware part of the Simulink model, defining the interface signals to be mapped on the various pins available on the FPGA whereas the System Generator block fixes the co-simulation parameters. When the design has been completely created, the set of the hardware blocks can be mapped on a real FPGA board in order to evaluate the percentage of used resources and the behaviour by hardware/software co-simulations. This is very useful to accelerate the simulation time compared to the totally software case.

The $2.048MHz$ sampling frequency was generated using a clock enable that delays the significant edges of the board clock signal in order to handle the synchronization of the digital decimation filter with the output signal coming from the software analog blocks. The designed decimation filter was tested on a Xilinx FPGA Virtex-5 LX330 but the hardware implementation can be mapped even on a smaller Spartan-3E 1600 development board, characterized by a lower amount of available resources, determining a slices utilization equal to 100%.

2.4 Simulation Results

The PSD of the output signal for a full-scale sinusoidal input allows to compute the Effective Number of Bits (ENOB) achievable by the converter. Considering only the sigma-delta modulator, without taking into account non-idealities effects and with an input at $2.7kHz$ of a $0.5V$ amplitude, the modulator grants to achieve a $SNR = 98dB$ corresponding to an ENOB of $16bit$.

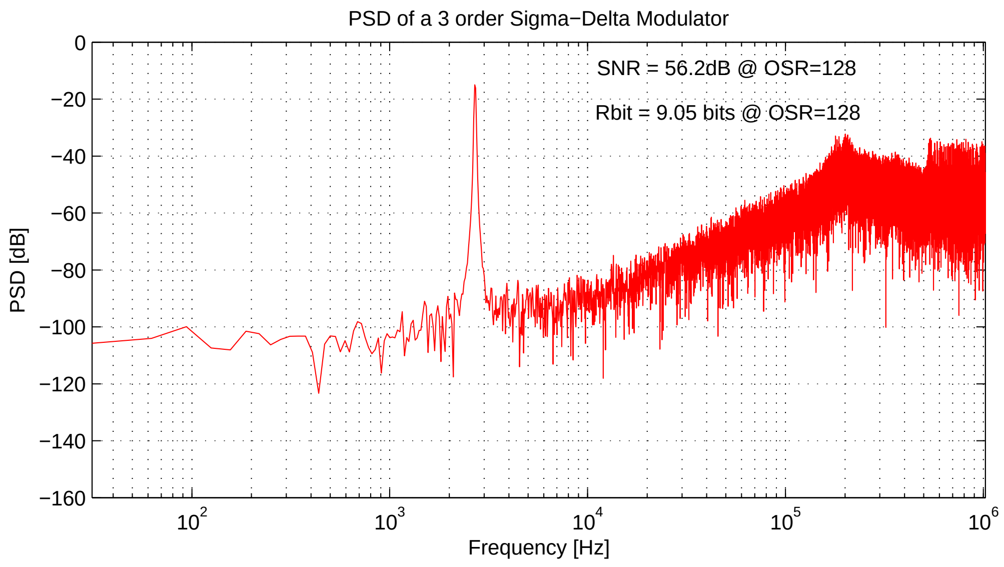


Figure 2.8: Sigma-delta modulator: Power spectral density of the complete real model

PSD results obtained simulating all the recording chain (filter + modulator), including also all the noise sources, show a deep degradation in terms of resolution, leading to a $SNR = 56dB$ corresponding to a $9bit$ converter resolution (Fig. 2.8). The plot was obtained using an input with an amplitude of $2mV$ at $2.7kHz$. It should be clear that this is a worst case condition, in which the overlapped noise has been extra-estimated, while in the real implementation it will be expected to have better noise performances.

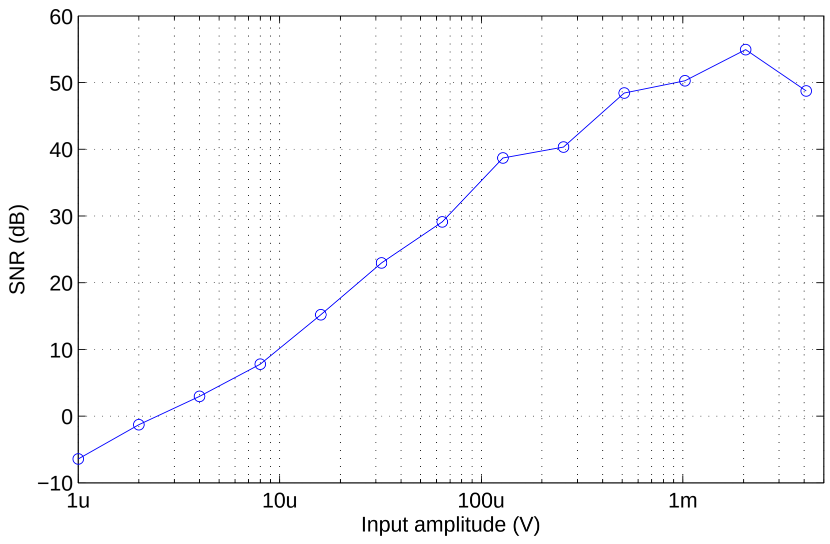
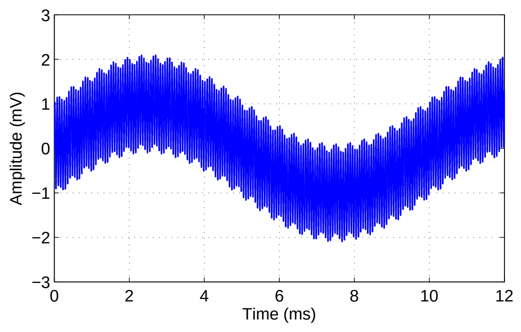


Figure 2.9: Signal to Noise Ration variations with different signal amplitudes

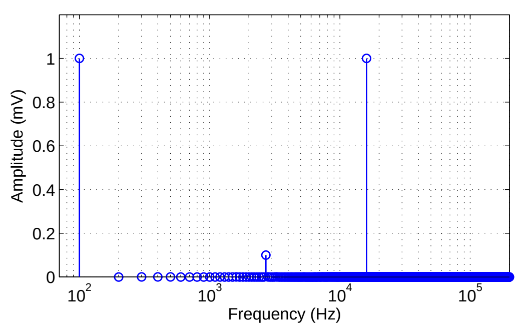
Fig. 2.9 depicts the system response in terms of SNR for increasing input amplitudes. As expected, the SNR increases for increasing input amplitudes and starts to degrade when a $2mV$ input amplitude is reached. This value corresponds to a modulator input of $0.5V$, i.e.

$V_{ref}/2$. The sigma-delta modulator performances drop drastically [77] because of saturation. It can be observed that the minimum detectable signal (corresponding to $SNR = 0dB$) is equal to $2\mu V$ which is an acceptable result for the signal of interest.

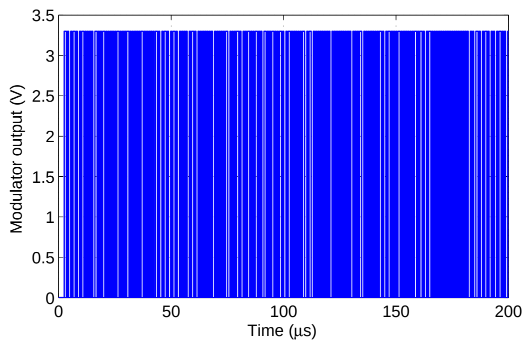
Transistor level design was validated by simulating the whole recording chain with an input given by the sum of 3 sine waves at 3 different frequencies. A $100\mu V$ signal at $2.7kHz$ has been used to emulate the neural signal, while two components with an amplitude of $1mV$ at $100Hz$ and $16kHz$ have been used to model out-of-band interferences. In particular, the low-frequency sine wave represents a realistic EMG signal. The input signal is shown in Fig. 2.10(a) and Fig. 2.10(b). The signal has been pre-filtered and pre-amplified by the analog block before being converted in a $1-bit$ stream by the sigma-delta modulator. The resulting signal is shown in Fig. 2.10(c) and its PSD can be observed in Fig. 2.10(d). The 3 components are still detectable and, as expected, the $2.7kHz$ component has been amplified more than the two out-of-band signals. The noise shaping effect due to the delta delta modulator is also evident. In order to remove the unwanted components, the signal has been decimated with a 32^{nd} order IIR Butterworth filter mapped in the Xilinx FPGA Virtex-5 LX330. The results in the time and frequency domain are reported in Fig. 2.10(e) and Fig. 2.10(f); it is evident how the unwanted components are completely filtered out and only the in-band frequency, amplified by the $45.5dB$ filter gain, is allowed to pass. The results obtained with transistor level simulation are then exactly what expected, confirming the high reliability of the developed behavioural model.



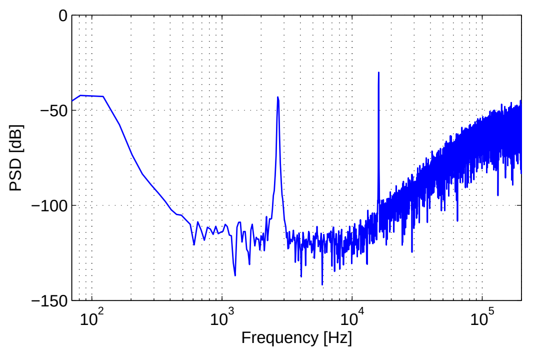
(a) Input Signal



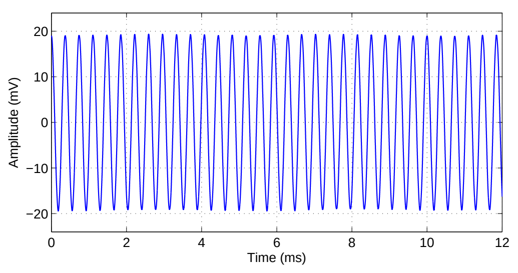
(b) Input Spectrum



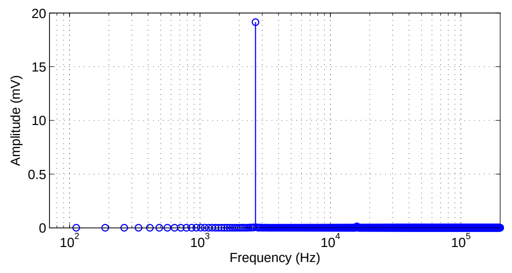
(c) Modulator Output Signal



(d) Modulator Output Spectrum



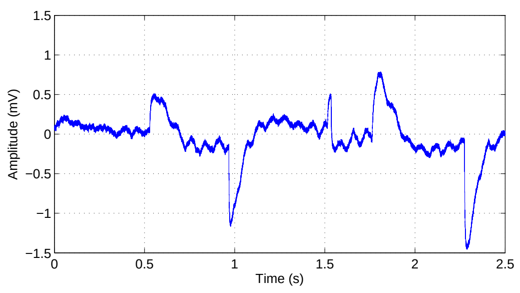
(e) Output Signal



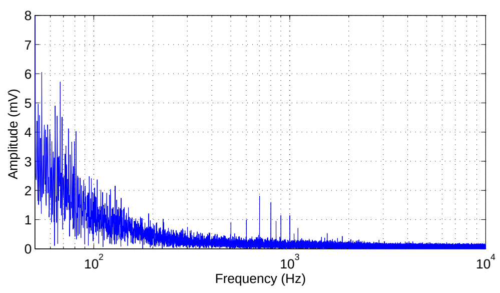
(f) Output Spectrum

Figure 2.10: Recording chain response to an input composed by the sum of three sines at $100Hz$, $2.7kHz$ and $16kHz$

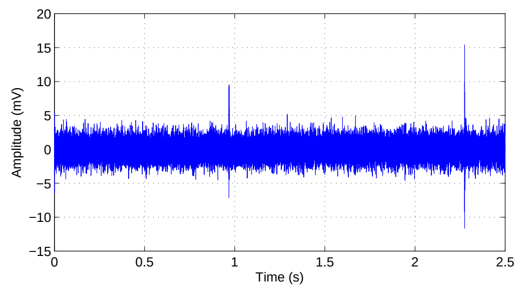
Once that the proper behaviour of the Simulink model has been confirmed comparing short-simulation time results with those achieved in Cadence environment, longer simulations can be reliably run using the more lean and flexible Simulink model.



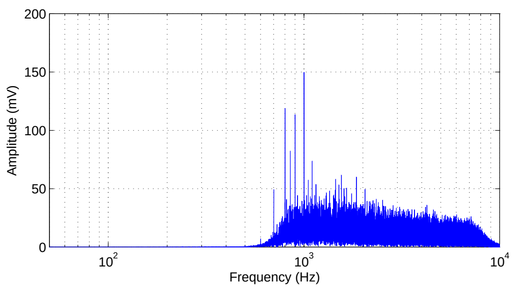
(a) Input Signal



(b) Input Spectrum



(c) Output Signal



(d) Output Spectrum

Figure 2.11: Sigma-Delta modulator: pre-recorded neural signal processing

In order to evaluate the system capability to work with real neural signals, the whole system has been tested with a pre-recorded neural signal extracted in clinical trials from the PNS of a rabbit subjected to cutaneous afferent stimulation at $50Hz$ and $100Hz$. The signal was filtered and modulated by the Simulink model of the analog module and the resulting stream of bits was fed to the IIR decimator filter mapped on the same FPGA Xilinx Virtex-5 LX330. The input pattern, represented in Fig. 2.11(a), corresponds to 2.5 seconds of recording. The input signal is affected by EMG and ECG interferences with a very large amplitude (in the millivolts range) and a spectrum (Fig. 2.11(b)) concentrated below $300Hz$. Such interferences completely mask the underlying neural content.

Fig. 2.11(c) displays the input-referred output signal and Fig. 2.11(d) its power spectral density: the low-frequency interferences are completely removed and the weak neural signal is now amplified and visible, as well as its frequency signature. Some tests have been also performed by processing a part of the track recorded in absence of external stimulation and, as expected, no evidence of neural spikes was detected and only the underlying noise was visible at the system output. In this way, it has been verified the functionality of a neural recording interface based on a sigma-delta architecture which can be exploited for the development of a real-time hardware implementation able to record multi-channel peripheral neural signals, working even in case of critical conditions.

Chapter 3

A Multi-Channel Electronic Interface for PNS Recording and Stimulation

3.1 Introduction

The human brain and the way it communicates with the rest of the body has always fascinated the researchers of all the world. The possibility to extract the neural signals and to use them outside of the human body in order to control a prosthetic device and re-establish the functionalities lost by a patient subjected for example to an upper-limb amputation is, in fact, one of the most challenging ideas of ever. To perform this task, the relative solution depends basically on the place where the signals are extracted. In literature there are a lot of works concerning the CNS neural recording systems [87, 21, 78, 83, 44, 28, 84], while a fewer number of works have been published related to the PNS signal acquisition [57, 17, 48, 50].

This is partially due to the wide range of possible applications offered by a CNS implant, being the brain the part of the body in which each neural signal is generated, this solution can be in fact exploited for many diseases related to the nervous system such as Epilepsy, Alzheimer, Multiple Sclerosis, damages to the spinal cord and neuroprosthetics [29]. Moreover with a CNS interface it is possible to acquire different types of signals: in particular the Local Field Potential (LFP) and the Extracellular Neural Action Potential (ENAP) [21, 27]. Nevertheless such kind of implant is highly invasive and implies the electrode insertion inside the skull. An alternative way to reach the neural fibers is the implantation of electrodes in the peripheral nerves, such a solution is less flexible and more focused on a specific disease than that based on the CNS, but has the great advantage to be less invasive and recent studies show that it allows to achieve excellent results in the decoding the intentions of movements for prosthetic applications [57], especially using multi-channel intra-fascicular electrodes [63].

The general trend in neural recording is to acquire the signal by using multi-channel electrodes in order to discriminate the significant electrical activity of single axons and to evaluate the correlation between the signals coming from near recording sites. In literature many systems with a large number of electrodes have been presented, especially referring to the CNS systems extracting the signals of interest by Multi-Electrode Arrays (MEAs) characterized by a number of electrodes up to 256 [87, 21, 12]. The number of channels explosion has a deep impact also on the chip area in the perspective of a low-power real-time implementation responsible to the recording phase, that should not exceed the size of few mm^2 to allow an easy implantation.

All these considerations are still true considering the PNS but in this case the constraints are less stringent due to the fact that these electrodes have typically 8-16 channels. The main concern in PNS recording is to amplify the neural signals, whose typical amplitudes are in the tens of microvolt range, as specified in the previous section, filtering the electro-stimulations signals (EMG interferences) of the muscles which surround the electrodes whose spectrum partially overlaps the bandwidth of the signals of interest and with amplitudes in the range of millivolt. For these reasons, the recording module front-end must be designed with a special care aimed to amplify the weak neural signals avoiding the risk of amplifier saturation due to the huge EMG interferences. At the same time, the stimulation unit plays a key role in neural interfaces, since it permits to restore the sensory feedback to the patient giving information about the external world. The sense of touch can, in fact, be emulated by special sensors collocated in the cybernetic hand and activated by pressure, and the picked up information can be transduced in electrical currents to be injected in the patient's stump with the same electrodes used in the recording phase with amplitudes and morphologies strictly dependent on the information which must be sent to the brain, usually encoded though a neuromorphic model [20].

In this part of the research activity a bidirectional multi-channel neural interface, intended to record neural signals picked up from an electrode implanted in the PNS and to elicit the neural fibers with bi-phasic current pulses generated by a stimulation module, has been implemented. In particular, in this way, it is possible to send the peripheral neural signals to an external host processor in order to perform the next processing steps at off-line or to an electronic hardware platform able to decode the intentions of movements in real-time and give the correspondent electro-mechanical stimulations in input to a prosthetic device.

3.2 System Architecture

The system, as represented in Fig. 3.1, is aimed to permit the communication between the PNS of an amputee patient and a robotic limb in order to re-establish the functionalities lost by the former. It includes 8 parallel recording channels that are based on sigma-delta A/D converters following the architecture described in the previous chapter 2, and 8 independent stimulators, implemented as current D/A converters. The architecture has been developed on two different levels: an analog front-end and a digital back-end. The analog part has been implemented in an Integrated Circuit (IC) designed in a $0.35\mu m$ CMOS process from AMS (Austriamicrosystems) with double-poly capacitors and 4 metal layers while the digital back-end is hosted in the Xilinx FPGA Spartan-3E 1600 development board.

3.2.1 Analog Front-End

The analog front-end includes all the devices that directly interact with the implanted electrode. The system is capable of recording 8 channels in parallel, nevertheless, it has been thought to work with 16 channel electrodes, therefore an internal switch network has been introduced to allow choosing 8 out of the 16 available channels. The recording path is composed by a 1^{st} order Switched Capacitor (SC) band-pass filter (BPF) with programmable gain and a 3^{rd} order sigma-delta modulator with an oversampling ratio (OSR) of 125. It slightly

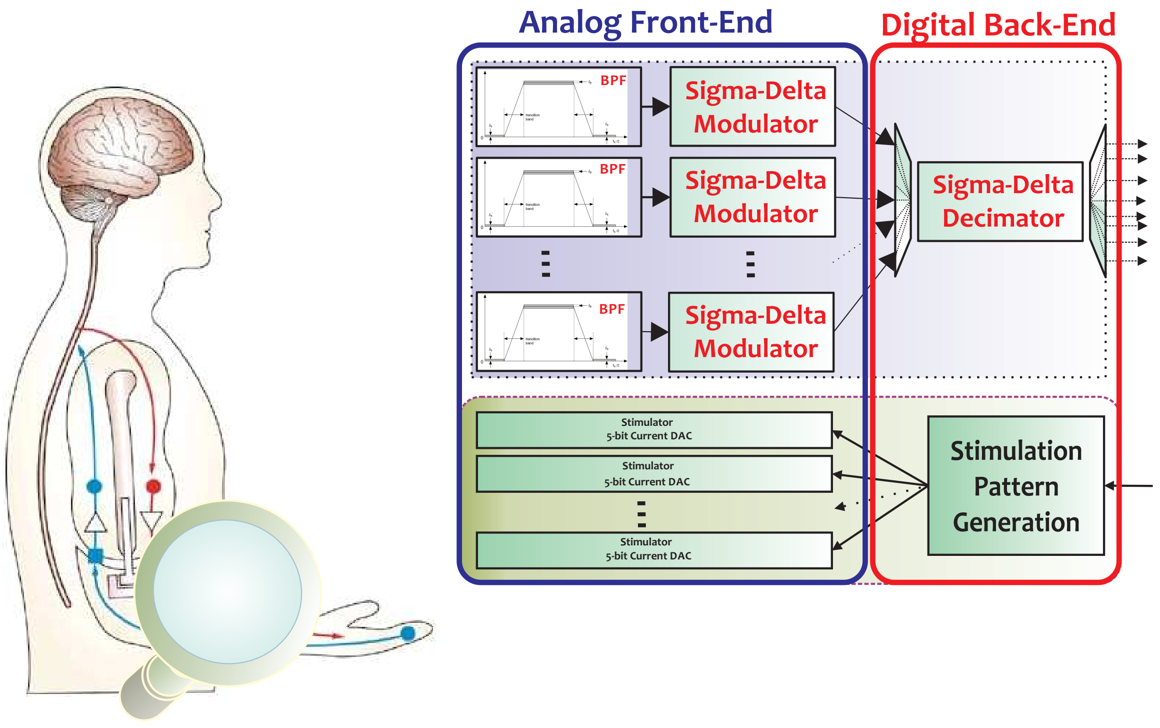


Figure 3.1: Block diagram of the neural recording and stimulation system

differs from what presented in the previous chapter in order to obtain an integer factor between the FPGA clock signal frequency and the oversampling frequency used for the sigma-delta A/D converter. In this way, considering an useful bandwidth for the signal of interest in the range between $800Hz$ and $8kHz$, the input oversampling frequency for each channel must be equal to $2MHz$.

Concerning the analog BPF, its main aim is to filter the neural signal from noise and unwanted biological interferences such as the low-frequency EMG interferences and to amplify it as close as possible to the site where the signal has been extracted, avoiding signal corruption. Due to the strict area constraints, to allow the implementation of an implantable device, it has been chosen a low order solution shifting the filtering complexity to the digital side exploiting the greater integration capacity. The gain must be set properly to benefit from the amplification effect without risking the loss of information due to amplifier saturation. To prevent this risk, the gain can be modified in a $46dB - 56dB$ range according to the actual input amplitude. The analog signal is then converted into a digital one using a 3^{rd} order sigma-delta modulator, the two level quantizers generate 8 single-bit streams at the oversampling frequency that are encoded using an SPI (Serial Peripheral Interface) protocol and sent to the FPGA to be processed by the high-selective sigma-delta decimator. The stimulation unit is a 5-bit current D/A converter, it provides a bi-phasic current that can range from $24\mu A$ to a maximum of $384\mu A$ with impedances up to $4.8k\Omega$, for higher values the maximum reachable current value scales down proportionally. The layout of the chip implementing the multi-channel analog part for recording and stimulation is reported in Fig.3.2(a) together with its view in Fig.3.2(b) obtained using a common electron microscope, the total area occupation is $4.1mm \times 4.1mm$.

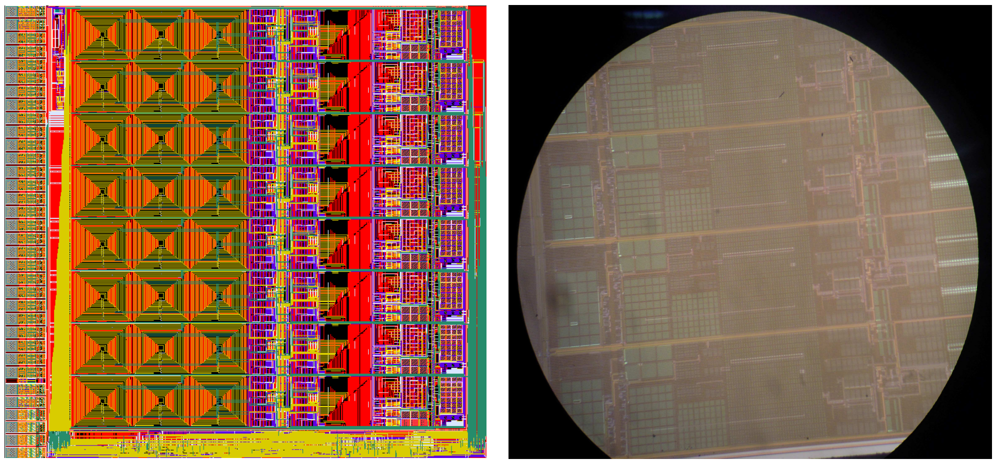

(a) Layout (b) Electron Microscope view

Figure 3.2: Eight-channels analog front-end mapped on the custom designed IC.

3.2.2 Digital Back-end

The main task of the digital part is the implementation of the sigma-delta decimator. As detailed in the previous chapter, it is typically composed by two main stages: a low-pass anti-aliasing filter and a downsampler. The former reconstructs a multi-bit signal from the one-bit stream generated by the modulator, removes the quantization noise shaped at high frequencies and preserves the signal from aliasing effects before the downsampling process in which one sample every R is picked up to bring back the signal to the Nyquist frequency relaxing the real-time constraints for the next steps of , especially in the case of a peripheral neuroprosthetic device. The R factor is fixed to 125 and must coincide with the oversampling ratio (OSR) of the sigma-delta converter. Since, in this specific application, one of the most critical design issue is the low-frequency EMG interferences rejection, the digital filter has been designed as a high-selective band-pass filter (BPF). In particular, a BPF in a $800Hz - 8KHz$ bandwidth has been implemented using a 16^{th} order Infinite Impulse Response (IIR) Butterworth solution that allows obtaining a good compromise in terms of out-of-band attenuation and necessary hardware resources utilization as it has been demonstrated performing hardware/software cosimulations exploiting the Xilinx System Generator tool in Matlab Simulink environment. In this case, a lower order has been chosen in comparison to what described in the previous chapter due to the fact that the target FPGA must host a more complex digital control system to manage the various phases of recording and stimulation and to test in real conditions the performance of the proposed solution.

After verifying the correct operation of the IIR filter on Matlab Simulink using a double-precision floating-point representation of the internal signals, the same functionality has been evaluated in the case of fixed-point for which it is necessary to specify the number of bits and the correspondent fraction length. The main filter parameters for the fixed-point representation are summarized in Table 3.1. Regarding the input word length and the relative fraction part, the values have been chosen considering that the recording system must be able to detect neural pulses with amplitudes in the order of tens of microvolt. The risk of overflow for the integer parts of the internal signals is prevented thanks to the fact that the obtained coefficients have values less than unity.

Response Type	Band-Pass Filter
Design Method	IIR Butterworth
Filter Arithmetic	Fixed-Point
Passband	0.8 - 8 KHz
Input Sampling Frequency	2 MHz
Order	16
Coefficient Word Length	32 bit
Input Word Length	32 bit
Input Fraction Length	22 bit
Output Word Length	32 bit
Rounding Mode	Floor
Overflow Mode	Wrap

Table 3.1: Digital filter parameters

The HDL code corresponding to the simulated 16^{th} order quantized IIR filter has been developed in order to verify the correct functionality when it is mapped on a real FPGA. Matlab *FDATool*, exploited for the design of the band-pass filter and to verify the functionality in the target application at simulation time, determines the architecture of the desired filter as a cascaded of 2^{nd} order *Biquad* sections using a pipeline mode in order to maximize the throughput of the digital filter. The pipelined structure is simply created positioning a delay register between two consecutive subsections.

Once that the filter performances on one channel have been verified, the results have been extended to a multi-channel implementation. A solution based on a common Time Division Multiplexing (TDM) technique has been adopted in order to exploit the same hardware resources for each channel. As shown in Fig.3.3, the resultant block diagram of the multi-channel digital filter has been implemented in which the various delay registers present in the single-channel solution and located in each Biquad section are replaced with banks of 8 registers used as circular buffers according to the current value of a 3-bit counter incremented at the significant edges of a clock signal of frequency $8 \times 2MHz = 16MHz$. For this purpose, a sampling frequency of $16MHz$ (that is eight times the value necessary for each channel) has been used. Synthesis results in terms of slices utilization on the target FPGA and area occupancy have also been calculated in the perspective of a low-power ASIC implementation.

Table 3.2 compares the results achieved with the TDM method with a static solution that simply replicates eight IIR filters. In the latter case, the actual ratio in FPGA synthesis would be greater than 100% making the implementation through TDM a mandatory solution for the target device. It should also be noticed that the number of the embedded multipliers used for the multi-channel solution of the digital filter is less than the corresponding number for the single channel version because some of them have been synthesized as logic blocks. These results have been obtained exploiting the Xilinx Synthesis Tool (XST) and mapping the hardware implementation on a Xilinx FPGA Spartan-3E 1600 while the ASIC results have been derived through a commercial release of the Cadence SoC Encounter software with a *90nm* low-power technology library.

The digital module is also responsible of generating the stimulation patterns which must be given in input to the same electrodes used in the recording phase based on external in-

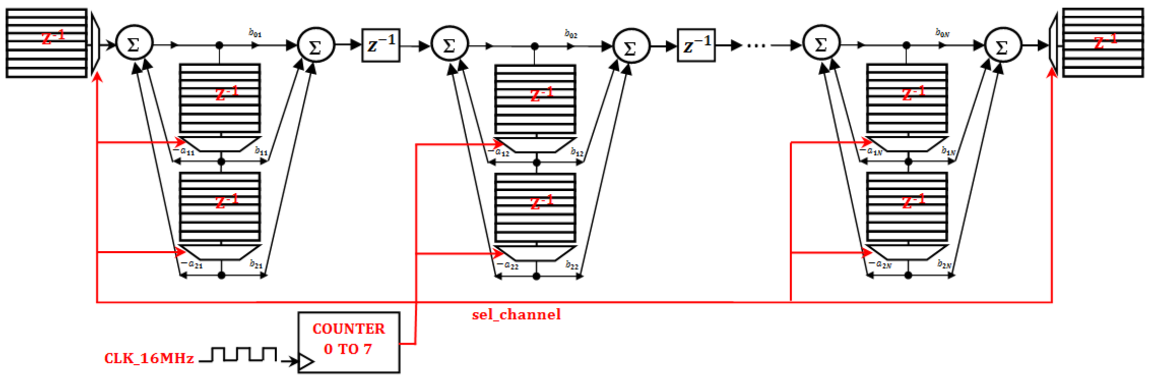


Figure 3.3: Block diagram of the multi-channel version of the digital BPF.

FPGA synthesis	**8 single ch.**	**8 mux. ch.**
Number of slices	39656/14752	9374/14752
Number of slice flip flop	6432	4933
Number of 4 Input LUTs	76768/29504	14941/29504
Number of MUL18x18SIOs	288/36	32/36
Actual ratio [%]	268	63
Maximum frequency [MHz]	40.476	30.878
ASIC synthesis	**8 single ch.**	**8 mux. ch.**
Cells	135016	27095
Cell Area [μm^2]	1083624	255439
Leakage Power [μW]	790.38	161.32
Dynamic Power [mW]	24.08	12.09
Total Power [mW]	24.87	12.25

Table 3.2: Synthesis results in the case of 8 parallel channels (column 1) and in the case of 8 multiplexed channels (column 2)

formation extracted by particular sensors placed in the surface of the cybernetic hand and in particular to define the pulse parameters in terms of pulse duration, frequency and amplitude as it will be described in the next sections.

3.3 Testing Environment Setup

To verify the functionalities of the implemented bi-directional neural interface in recording and stimulation, firstly from the electrical point of view and secondly by means of in-vivo experiments on sedated animals in real critical conditions, a dedicated test environment has been developed. Fig.3.4 highlights the relative structure which is mainly divided into two parts: a custom designed Printed Circuit Board (PCB) and a Digital Control System (DCS) mapped on a Xilinx FPGA Spartan-3E 1600 development board.

In Fig.3.5 a picture of the testing system is shown, involving the Xilinx FPGA Spartan-3E 1600 Development board on the left side and the custom designed PCB on the right one that are linked through a 100-pins Hirose connector on which are routed the various signals

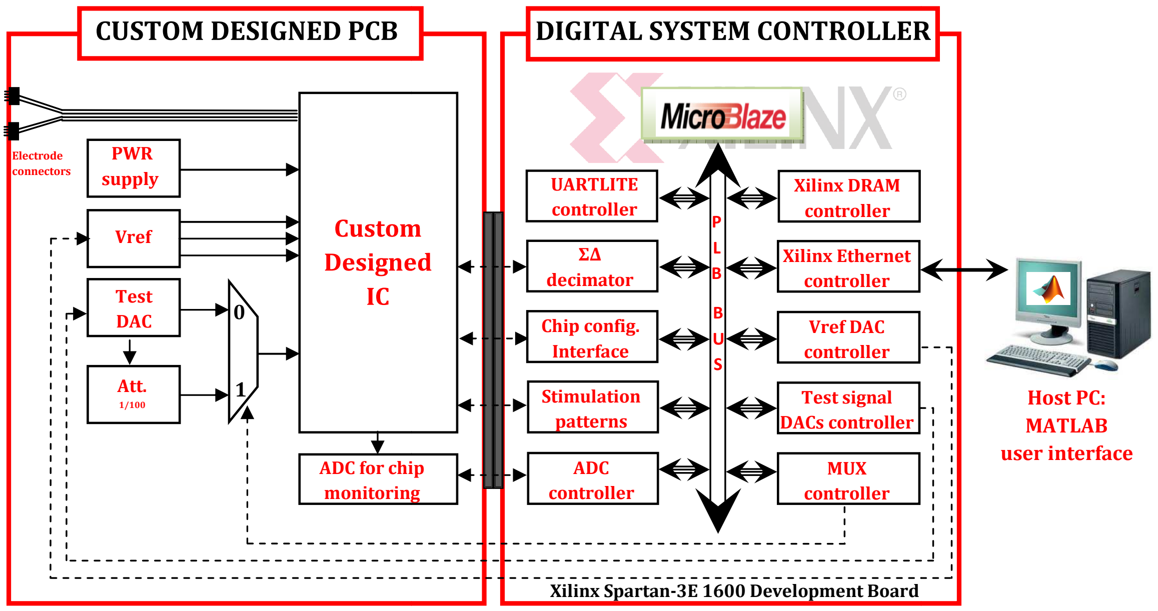


Figure 3.4: Architecture of the Digital Control System.

needed to allow the communication between the two parts. In the following both will be described in details before presenting the significant results of the neural interface in the conditions of interest.

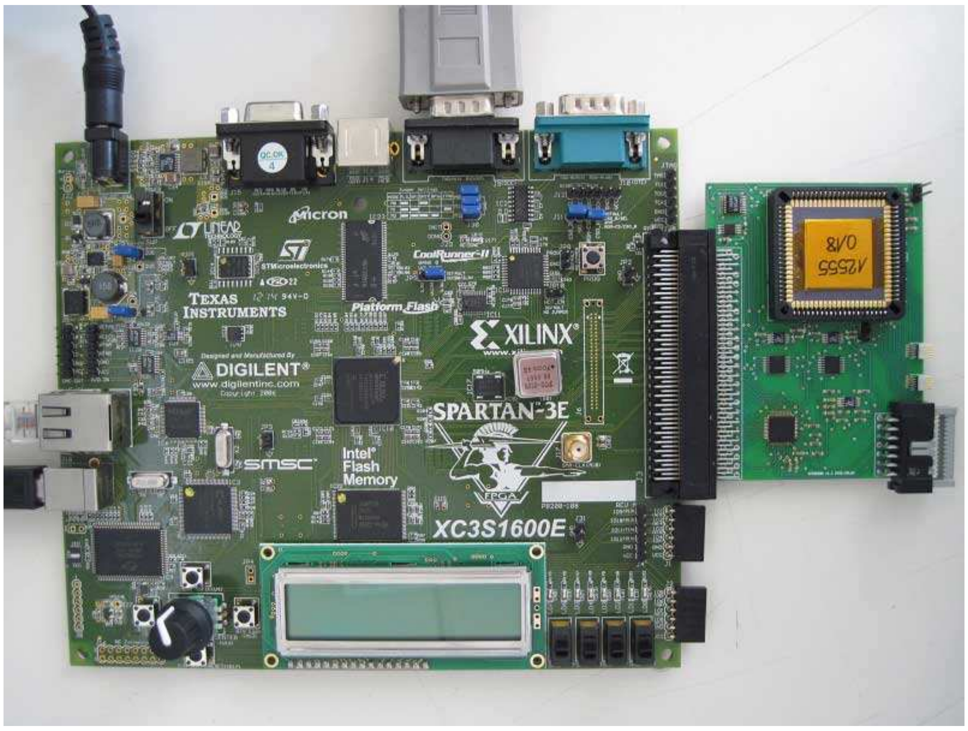


Figure 3.5: Implementation of the Testing Environment.

3.3.1 Custom Designed Printed Circuit Board

The PCB has been designed using the software Cadence-Orcad Capture for the schematic and PCB Editor for the Layout. Special care has been put on the latter, trying to minimize the connections and keeping as symmetrical as possible the tracks carrying the most critical analog paths. Wide ground planes have also been used as well as large tracks for ground and power supply paths.

The main aim of the PCB is to host the ASIC recording and stimulation chip and all the devices needed to provide power supply and reference voltages to the chip, as well as to generate the analog input signal needed to verify its behaviour in the test conditions. In particular, it has been used a $3.3V$ voltage regulator (component MAX1792 from *Maxim Integrated*) for the analog power supply generation. Different power domains have been adopted for the digital and the analog parts to avoid that the huge digital noise is picked up by the analog circuits which is more sensitive compared to the previous one.

The digital devices have been supplied with a $3.3V$ provided by the Spartan-3E 1600 board and transmitted to the PCB through the Hirose connector. The ASIC chip requires three different reference voltages for its working: a $1.65V$ as reference voltage in all the switched capacitor circuits and two voltages ($0.65V$ and $2.65V$) for the sigma-delta V_{ref} generation. They are provided by a voltage DAC (component LTC2604 from *Linear Technology*), this choice allows tuning the reference voltages and to adjust, by this way, the converter resolution if needed during the test phase thanks to the reconfigurability allowed by the digital control system.

The fully-differential input signals for the chip testing are generated by two 16-bit DACs (component LTC2641 from *Linear Technology* has been used), achieving a Least Significant Bit (LSB) of $50\mu V$. Since the typical neural amplitudes are in the decades of microvolt range, an attenuator has been cascaded to the DACs to allow the system test with more realistic signals. A low-noise fully differential operational amplifier (component LT1994 from *Linear Technology*) with a proper resistive feedback network has been used to implement the attenuator. Thanks to a switch network (component ADG636 from *Analog Devices*) it is possible to select if use as system input the DACs or the attenuator output. The PCB hosts also two 12-pin connectors for the electrodes connection to test the system during the in-vivo experiments on animals. An ADC (component ADS5560 from *Texas Instruments*) has also been introduced to monitor the behaviour of the intermediate stage in the recording channel.

3.3.2 Digital Control System

The Digital Control System (DCS) is aimed, on one hand, to configure and manage the ASIC chip and all the devices hosted on the PCB and, on the other hand, to handle the communication with an host PC on which the next processing and decoding steps can be performed before the realization of a fully portable implementation of the neuroprosthetic device. In this way, the user can require the execution of a particular test, to record in real-time the various outputs of the system architecture mapped in the chip during the recording phase or to send bi-phasic stimulations with a specified set-up in input to the same electrodes used in recording by exploiting an user-friendly Matlab interface.

The DCS is based on the use of a MicroBlaze processor mapped on the target FPGA which represents a soft-core used as the micro-controller of the system and the intermediate stage which allows the communication between the user and the chip, especially to configure its behaviour in adaptive way according to the recording conditions. It involves several peripheral modules devoted to control the different devices placed in the PCB, as shown in fig. 3.4. The communication between the modules and the processor is performed through a Processor Local Bus (PLB) by which the MicroBlaze can assess on them by a memory-mapped solution.

In particular, besides the sigma-delta decimator that is an integral part of the recording unit (described in section 3.2.2) responsible of cutting the low-frequency interferences and avoiding aliasing effects before the downsampling process at the Nyquist frequency for each channel, the DCS hosts several hardware peripheral controllers. Regarding the devices on the PCB it contains the controller for the chip configuration and for the stimulation patterns definition, as well as the controller for the DACs that generate the reference voltages, for those used to generate the input signals, for the ADC and for the switches configuration. It includes also a control module for the Ethernet communication with the host PC, a UartLite controller for the MicroBlaze debug prints and one for an external DDR DRAM memory in which it is possible to save several amount of data exploiting its greater capacity and to map the instruction code and the data structures of the MicroBlaze application. The DCS has been designed exploiting the Xilinx Platform Studio (XPS) tool for the creation of the embedded system at higher level which must be mapped into the target FPGA, specifying the hand-coded HDL implementation of each hardware controller after the correct evaluation of its behaviour at simulation time.

In this way, the user can ask the following operations during the electrical or the in-vivo experimental tests:

- set the value of the internal registers instantiated in the digital interface of the custom designed IC;
- generate configurable stimulation patterns as bi-phasic pulse trains;
- evaluate one of the intermediate outputs of the analog recording path;
- create fully-differential sinusoidal test signals with variable amplitude and frequency which can be applied in input to the designed IC and processed even by the digital part to verify the resultant frequency response of the system in the band of interest and the out-of-band attenuation, enabling or not the attenuator cascaded at downstream of the two DACs;
- acquire in real-time the samples coming from the different channels of the sigma-delta modulators and filtered by the decimator in order to allow an on-line recording and processing or a plot on a graphical user interface (GUI).

The hardware controllers, connected to the PLB, are accessible by the MicroBlaze through read/write operations on defined registers specified by an univocal address. The communication between these controllers and the correspondent components on the PCB is usually based on an SPI protocol specified in the relative datasheets. In the following the solutions adopted for some of the modules are described with more details.

Chip configuration

The digital interface integrated in the chip is characterized by a bank of eight registers, each one of 16 bit. Seven registers encode the information concerning the chip configuration such as the gain value of the band-pass pre-filter, the input test set-up bypassing one of the various stages involved in the recording path, the stimulation current values, the electrodes selection. They can be either read or written. The last one is a read-only register used to store the eight 1-bit generated by the sigma-delta modulators which must be applied in input to the multi-channel sigma-delta decimator.

The communication between the custom designed IC and its hardware controller is based on a custom SPI link synchronized with respect to a clock signal at the frequency of $40MHz$ handled by a determined Finite State Machine (FSM) which controls each phase of the chip configuration. This clock signal is created by the oscillator available on the FPGA according to the system clock of the board at $50MHz$ and instantiating a Digital Clock Manager (DCM) module in the digital part which determines the output clock signal properly specifying the values used as multiplier and divider, and finally transmitting it to the chip through the Hirose connector. In the same way, the relative clock signals are created and sent in input to the other available components based on the specifics defined in the relative datasheets.

The protocol consists of 20-bit transactions as shown in Fig. 3.6 which involves that each communication pattern between the digital part mapped on the FPGA and the chip on the PCB is performed with a frequency of $2MHz$.

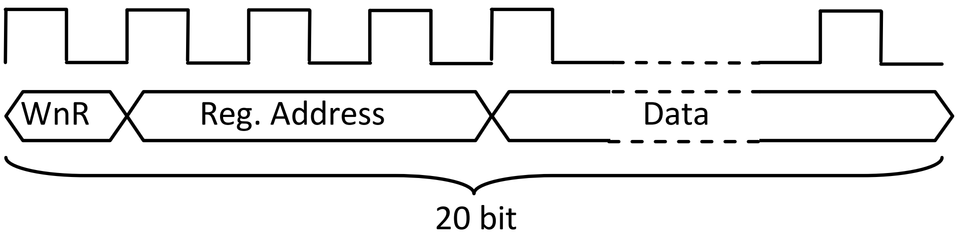


Figure 3.6: SPI communication protocol between the chip and the controller

In detail, the first bit indicates whether the operation is intended to read or to write one of the chip registers, the next three bits are used to specify the address of which register is involved in the operation and the last 16 bits are the data which have to be read or written on the register. Using this protocol it is possible to perform different tasks on the chip:

- “Reset” of the internal registers to their default values;
- “Writing” one of the 7 chip registers for the configuration of the analog front-end;
- “Reading” one of the 8 chip registers;
- “Streaming”, during which the bits generated by the eight sigma-delta modulators are continuously read at the frequency of $2MHz$ and sent to the digital decimator;
- “Stimulation” during which trains of bi-phasic pulses are generated as stimulation waveforms.

The proposed solution of the DCS allows to generate the train of bi-phasic pulses in stimulation mode and, at the same time, between two successive pulses, acquire the signal coming from the neural electrodes and processing them through the sigma-delta decimator. In streaming mode, the eight 1-bit outputs of the sigma-delta modulators are applied in input to the multi-channel decimator. They are filtered removing the unwanted low-frequency interferences and the quantization noise shifted at high frequencies, determining at output a better resolution able to properly reconstruct the signal of interest. Then the outputs are brought back at the Nyquist frequency, relaxing the timing constraints which must be satisfied in order to sent them in real-time to the host PC by the Ethernet link using the MicroBlaze processor as intermediate stage. This is performed by transmitting successive UDP packets with a payload containing a defined number of decimator output samples for the various channels represented with the correct fixed-point representation, which must be properly processed at PC side. To this aim, a buffering mechanism is required before to sent the samples in real-time to the PC via Ethernet. For this reason, the processor stores temporarily them on the external DDR DRAM memory. Special care has been in fact put on buffering operation in order to avoid the loss of samples due to saturation conditions. Due to the fact that the MicroBlaze must serve several operations at the same time, even the peripheral module responsible of the chip configuration saves the decimator output in a bank of 6 slice registers for each channel to further reduce the frequency, with respect to the Nyquist rate, according to which the processor acquires the relative signals before sending them to the host PC for the next processing steps.

As specified before, write operations on the internal registers allow the user to reconfigure at runtime:

- the gain of the various processing sub-stages of the analog front-end (high-pass and low-pass pre-filtering) for all the 8 channels;
- to give the test signal in input to one of the various stages in order to evaluate the frequency response of each of them or considering as input that coming from the neural electrodes for in-vivo experimental tests;
- to bypass or not the stage of bandpass pre-filtering;
- to set the value of the stimulation currents for each channel;
- to define which of the intermediate outputs of the processing flow connect to the test pins in order to verify the behaviour with an oscilloscope.

Stimulation pattern generation

The programmed stimuli are bi-phasic pulses (Fig. 3.7) in which parameters such as amplitude (A), width (W), frequency (f) and the number of pulses can be defined by the user at runtime within pre-defined ranges as indicated in the Table 3.3.

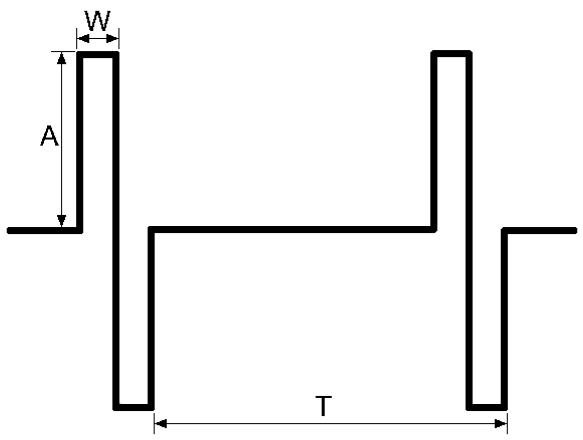


Figure 3.7: Biphasic neural stimulus waveform

parameter	range
A (μA)	$24-384$
W (μs)	$10-300$
f (Hz)	$10-300$

Table 3.3: Biphasic neural stimulus waveform

The stimulation pulse is generated by the hardware interface with the following operations handled by the same FSM dedicated to the chip configuration: first, the digital code corresponding to the selected amplitude A is written on the chip register, this values stays on the register for a time interval equal to W, then, the negative value is stored in the same register for the same time only reversing the relative sign bit and, finally, a recovery period with 0 current is set. The process is repeated after a period T equal to the inverse of the requested frequency f. During this recovery period, the operation state of the chip can be changed to streaming mode in order to continue the acquisition at output from the analog part of the recording path. Fig. 3.8 shows the set-up of the testing environment during the stimulation phase, in which it is possible to notice:

- the host PC by which the user can ask to the DCS, via the Matlab interface, the generation of a specified stimulation pattern;
- the test system involving the Xilinx FPGA Spartan-3E development board connect to the host PC via Ethernet and the PCB;
- a breadboard on which an equivalent resistive model of the input impedance of the neural electrodes are placed and connected to the relative pins available in the PCB;
- the MSO6054A mixed signal oscilloscope from Agilent technologies by which it is possible to evaluate the timing trend of the voltage drop across the resistive model and determine the stimulation current in input to the neural electrodes.

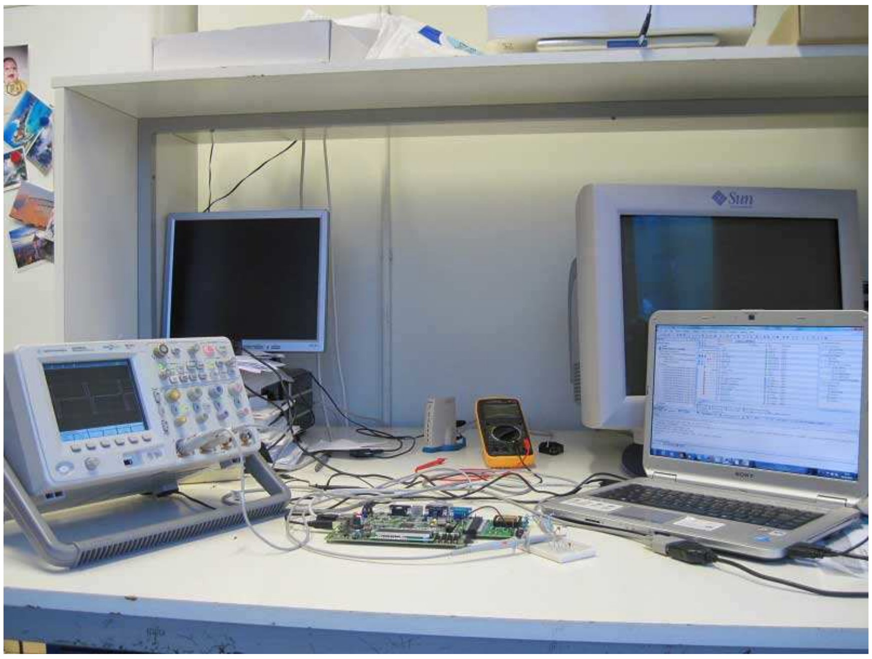


Figure 3.8: Set-up of the testing environment during stimulation.

Input test signal controller

The controller for the input DACs is aimed to generate electrical signals to test the filter frequency response. The DCS allows the user to select at runtime the type of the signal which must be applied in input to the chip, choosing among a fully-differential test signal generated by the two DACs and that coming from the neural electrodes. Regarding to the first solution, the user can choose between two different alternatives:

- to generate two sinusoidal waves with a phase shift of 180°and variable frequency and amplitude in order to verify the behaviour of the recording system for components along the whole band of interest and the attenuation in the case of out-of-band components;
- to apply input samples belonging to a real neural signal, recorded during previous in-vivo experimental tests on sedated animals and saved in the DDR DRAM memory accessible by the MicroBlaze processor.

Therefore, for the first test case, sinusoidal signals in a wide range of frequencies are required. Since the filter bandwidth is expected to be in the $800Hz - 8kHz$ range, the test frequencies cover an interval from $80Hz$ to $80kHz$. In such a wide range it is not possible to use the same number of samples per period for each frequency, therefore 3 different Look Up Tables (LUTs) have been used: 100 samples/period for frequencies below $4kHz$, 24 for frequencies in a $[4kHz - 20kHz]$ range and 6 samples for the highest frequency interval $[20kHz - 80kHz]$. This choice allows having a better resolution for low frequency signals considering that, for them, timing constraints are not particularly strict, for higher frequencies the number of samples is reduced in order to meet the constraint on the maximum input sampling frequency for the DACs. The two DACs receive the same samples with a 180°phase shift for the generation of a fully differential signal. To do this, the samples sent to the second

DAC start from half the LUT. For the second test case, the controller is also able to send to the two DACs generic samples coming from the PC, by this way, any waveform can be set as system input. To allow the generation of test signals with amplitudes in the order of tens of microvolt as for the signals of interest, the user can ask to the DCS of enabling or disabling the attenuator with a transformation ratio equal to 1/100 at downstream of the two DACs.

Ethernet controller

An Ethernet link has been used for the FPGA-PC link in order to exploit its great communication bandwidth and fulfil the timing constraints imposed by the application under consideration. In particular, the UDP protocol has been preferred to the TCP one for its lower latency. Indeed, the higher reliability of the TCP protocol is not necessary in this case, considering that there is a point-to-point connection between the host PC and the FPGA device.

LightWeight IP (LwIP) libraries have been used for the MicroBlaze code, they were included in an open-source project of C libraries for the implementation of the TCP/IP protocol stack optimized for the use on embedded systems. It can be used in many development environments for embedded systems including Altera, Xilinx and Honeywell. The mapping of such a system on the FPGA requires only few tens of kbytes of RAM and approximately 40 kbytes of ROM and has the great advantage of masking all the complexity at low level to handle the Ethernet communication. To do this, it has been also necessary to instantiate a Xilinx Ethernet controller, an Interrupt Handler, a hardware timer to handle the timing of the interrupt handler and the DDR DRAM to store the buffered packets before the sending and the code of the software library by properly configuring the Linked Script file for the XPS project of the embedded system. It assumes that in the configuring process of the target board, by using the XMD (Xilinx Microprocessor Debugger) tool, the ELF (Executable and Linkable format) file must be downloaded on the available external memory due to the fact that there is not enough space in the local memory of the processor, resulting in a higher latency access for read/write operations.

The optimization of the relative parameters can be made directly from the "Software Platform Settings" panel of the XPS tool: the few required operations to do are the maximum payload size definition for the UDP packets, the selection of the packets queue depth and the activation of the UDP communication. When the user sends a packet to the DCS, the MicroBlaze processor keeps track of its IP address and the used port creating the communication socket, invokes a specific callback function as soon as a packet is received, sends a response or performs further processing and finally verifies on the network interface if new packets are incoming. The type of required operation is encoded in the payload as a string.

In this way, the MicroBlaze acts as an asynchronous micro-controller for the DCS according to the user requests: at the packet receipt, it decodes the message in the payload, storing the significant information or delegating the operations to the dedicated controllers. The use of the LwIP library allows the processor to manage multiple requests at the same time coming from the PC. The latency introduced by the UDP protocol and the constraints imposed in terms of maximum payload size allow to perform a real-time communication only at the sigma-delta decimator output (i.e. 8 channels downsampled at $16kHz$). Nevertheless it is also possible to acquire the modulator outputs (i.e. 8 channels at $2MHz$), but in this case only an off-line transmission is possible. In this latter case, the number of samples that can be transmitted is limited by the space available on the DDR DRAM on which they are temporarily saved.

PC-side interface

From the point of view of the user application, an user-friendly interface in Matlab environment has been developed. It facilitates the Ethernet connection management since it handles the packet transmission and reception to and from the FPGA by running respectively simple write and read operation to the output buffer of the PC Ethernet interface. To do this, the communication socket with the board must be defined exploiting the "Instrument Control" toolbox.

An useful application is for example the feedback streaming audio generation to ear the neural recorded signals, this is very helpful during in-vivo tests. To perform such a task, custom scripts were created in order to read the information transmitted by the MicroBlaze, convert them in the proper format and save the samples on a buffer associated with the multi-channel audio interface of the host PC (through the instantiation of an *AnalogOutput* object). When a defined threshold is reached during the buffer filling, the script invokes an asynchronous callback whose task is to read and transfer the associated samples to the *AnalogOutput* object. By this way, the user can ear in real-time the signal and assess the presence of significant neural spikes.

3.4 Experimental Results: electrical tests

The first tests are aimed to verify the analog front-end behaviour from an electrical point of view. The test system based on discrete DACs described in section 3.3.2 has been used for this purpose. The PCB is provided with two test points used to monitor the chip during the tests: one is collocated at the chip output, a switch network allows selecting which intermediate output connects to them. In this way, it is possible to analyse the temporal trend at the HPF, the LPF, the 1^{st}, 2^{nd} and 3^{rd} modulator stage outputs observing them with an oscilloscope. The other test point is at the chip input and allows verifying the system performances during the stimulator tests.

3.4.1 Recording unit tests

Firstly, the frequency response of the analog 1^{st} order BPF has been verified, a differential sinusoidal input with fixed amplitude and variable frequency in the range $80Hz - 80kHz$ has been used. Fig. 3.9 shows the results, the blue curve was obtained with the lowest gain while the red one with the highest gain configuration.

The filter parameters, in terms of gain and bandwidth for the maximum and the minimum gain configuration, are reported in Table 3.4. It should be clear that, even though there is a slight difference with respect to the design specification ($800Hz$Âů$8kHz$) it does not represent a problem since the out-of-band frequencies will be completely rejected by the high selective BPF of the digital decimator stage.

Fig. 3.10 demonstrates the possibility to program the gain of the pre-filter: the traces were obtained stimulating the chip with a fully-differential sinusoidal signal with a $280\mu V_{pp}$ amplitude at $3kHz$. The weak amplitude has been obtained thanks to the attenuator cascaded after the two DACs generating the input test signal. The result has been achieved with 5 possible gain configurations; nevertheless, the gain can be configured with a total of 256 values between the minimum and the maximum values.

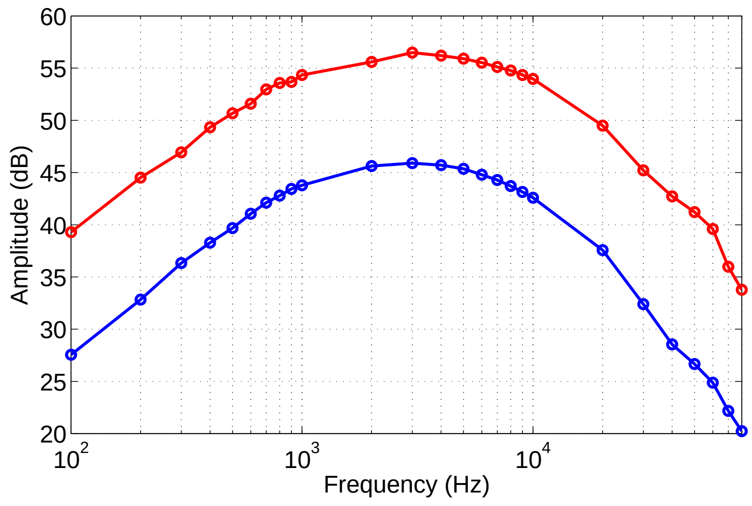


Figure 3.9: BPF Bode diagram. Red curve: higher gain configuration, blue curve: lower gain configuration.

	Gain [dB]	Bandwidth [KHz]
High Gain	56.5	0.8-11
Low Gain	45.9	0.8-9.5

Table 3.4: BPF parameters

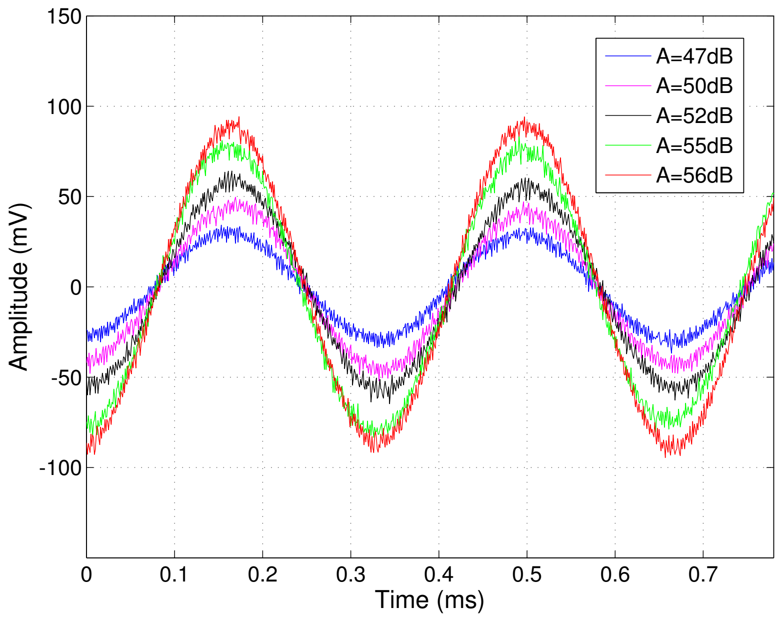


Figure 3.10: BPF: gain programmability.

As detailed before, the low-order band-pass pre-filter integrated in the custom designed IC for each channel is aimed only to partially remove the low-frequency interferences. It is due to the fact that then the greater integration capacity of the digital side and the proposed sigma-delta architecture have been exploited to design the first stage of the decimator with a highly selective frequency response, as can be seen in Fig.3.11, mapped on the target FPGA and, in the perspective of portable solution, hosted in the stump of the upper limb prosthesis device.

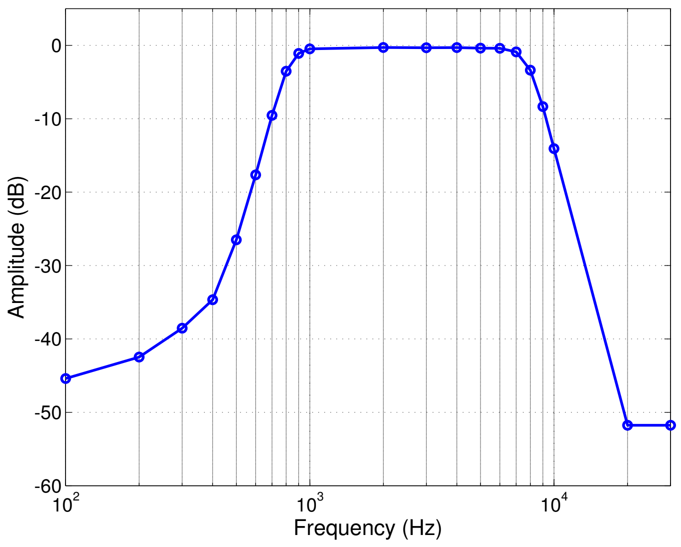
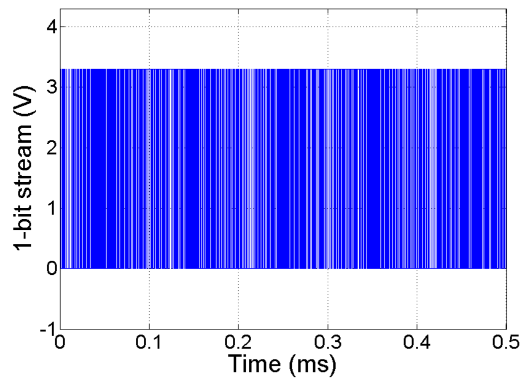


Figure 3.11: Frequency response of the 16th order digital BPF

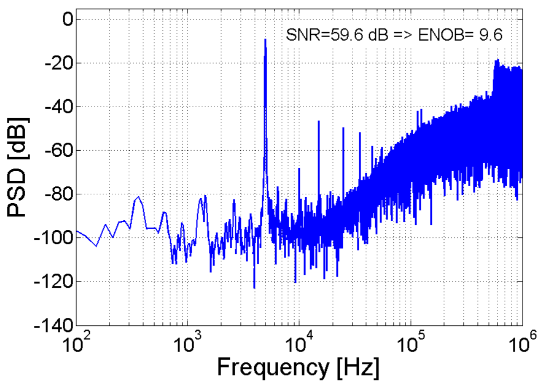
The sigma-delta converter resolution was tested setting as input a sinusoidal waveform of $0.5V@5kHz$, bypassing the analog BPF and applying it directly in input to the modulator. Fig.3.12(a) shows the 1-bit stream in the time domain while the resulting Power Spectral Density (PSD) is reported in Fig. 3.12(b) and confirms a predominant signal component at 5 kHz. The input amplitude corresponds to $V_{ref}/2$, it has been chosen to test the converter performances with a full-scale signal. In fact, for higher amplitudes the converter performances start to degrade. According to the resulting PSD it is possible to evaluate the signal-to-noise ratio that is equal to 57.1dB, corresponding to an Effective Number Of Bits (ENOB) of 9.2 bit. This can be considered the real resolution of the proposed converter. The plot points out also the noise shaping effect obtained by the sigma-delta modulator of the analog front-end even if this high frequency noise is then removed by the high-order digital decimation filter.

The digital filter output is reported in Fig. 3.12(c) and Fig. 3.12(d), respectively in the time and in the frequency domain. The digital decimator brought back the sample frequency to the Nyquist rate, the sine is thus sampled at $16kHz$. Therefore, it has only about 3 samples for each period, for that reason in Fig. 3.12(c) the spline interpolation of the measured samples has also been reported. The frequency domain shows how the high frequency noise has been rejected by the decimator filter. The digital BPF, in fact, is a 16th order filter and its sharpness allows attenuating deeply the out-of-band interferences, a $200Hz$ signal, for instance (that is a significant example being this frequency in the typical EMG frequency range) is attenuated by more than $100V/V$.

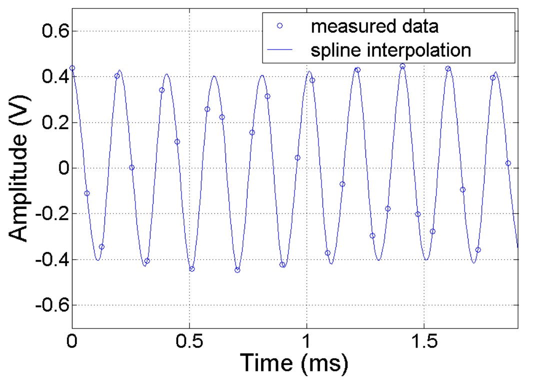
The relation between the signal amplitude (referred to the analog filter input) and the SNR calculated from the PSD at the modulator output has also been evaluated. The result is shown in Fig.3.13, a typical sigma-delta characteristic has been obtained, the SNR increases with the increasing of the input amplitude up to a threshold (in this case 3.6mV referred to



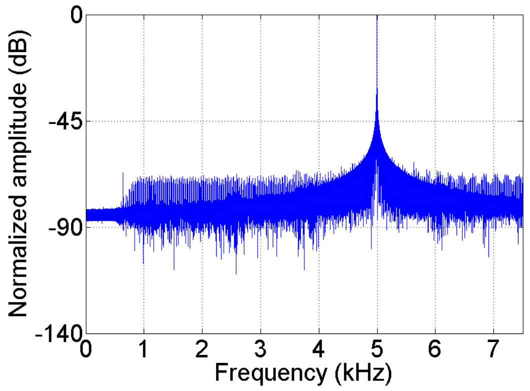
(a) Time domain: bit-stream



(b) Frequency domain: Power Spectral Density



(c) Time domain: digital decimator output



(d) Frequency domain: digital decimator output

Figure 3.12: Results at the modulator and at the decimator output with a sinusoidal input with $0.5V$ in amplitude at $5kHz$

the input) corresponding to about half the reference voltage at the modulator input. For higher amplitudes the signal saturates and the corresponding SNR starts to drop.

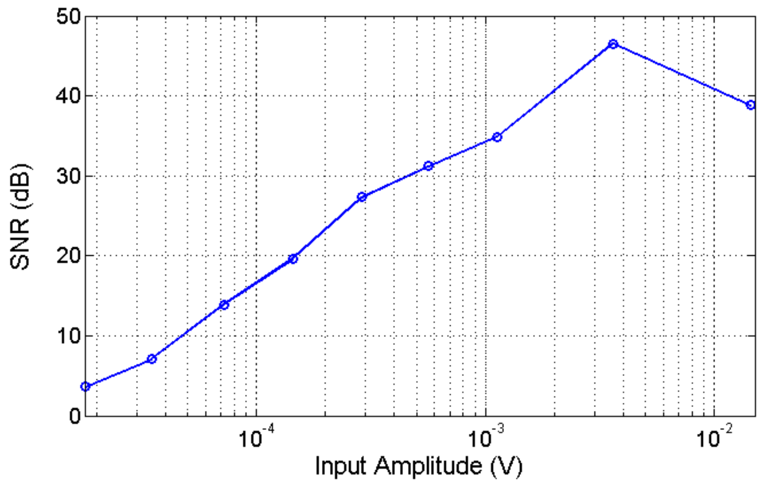
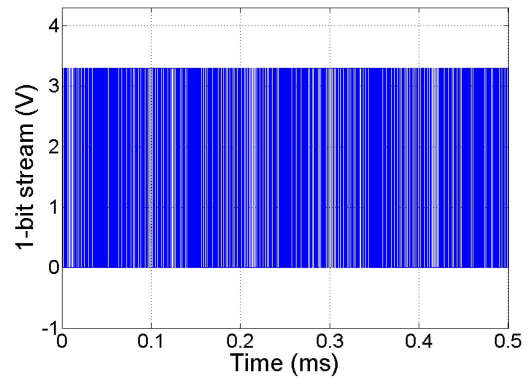
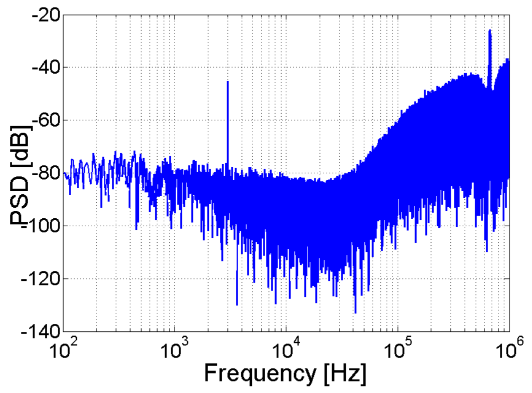


Figure 3.13: SNR vs. Input Amplitude.

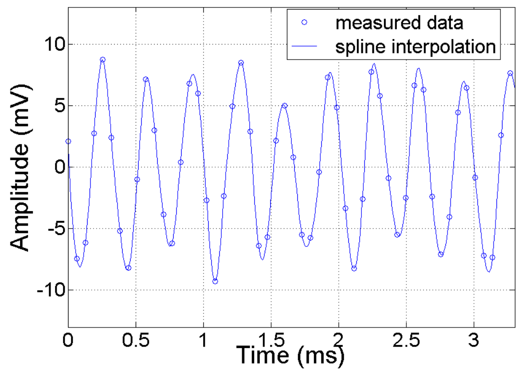
Since the system has been designed to work with neural signals, it has been also tested the capabilities of the whole recording chain to work with amplitudes in the order of tens of microvolts. A 18μV signal at $3kHz$ has been generated as input signal (the attenuator cascaded after the two DACs has been used for this purpose). In Fig. 3.14(a), the time 1-bit stream is reported, while Fig. 3.14(b) presents its PSD, even though such signal is particularly weak it is still detectable. This is even more evident if the output of the digital filter is analysed: Fig. 3.14(c) shows the acquired signal in the time domain, the result has been obtained using the higher gain configuration for the analog bandpass pre-filter in order to amplify the weak signal as much as possible before the digital conversion. The frequency spectrum, reported in Fig.3.14(c) confirms this result, showing a peak signal at $3kHz$ as expected. Looking at the underlying noise shape in Fig. 3.14(c), it is evident how the interferences below $800Hz$ are deeply attenuated.



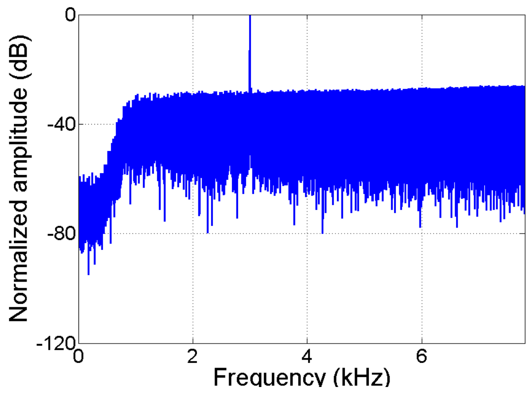
(a) Bitstream



(b) Power Spectral Density (PSD)



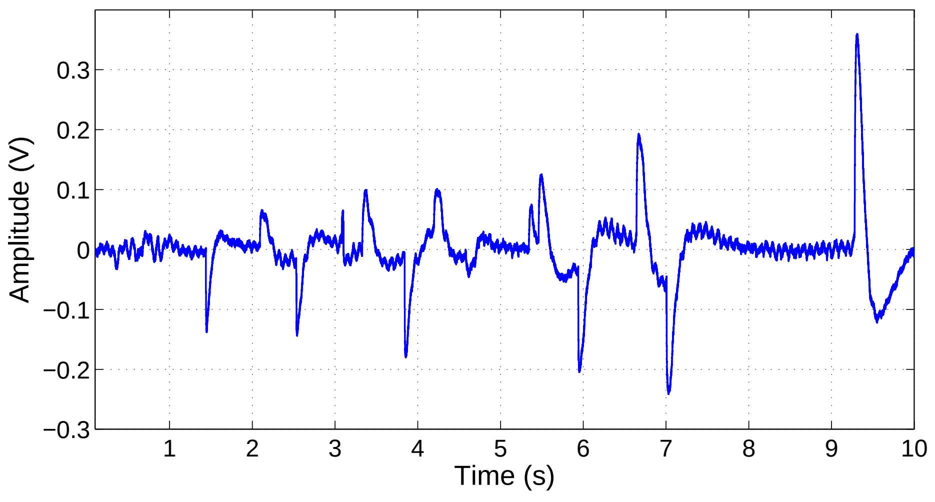
(c) Decimator output in the time domain



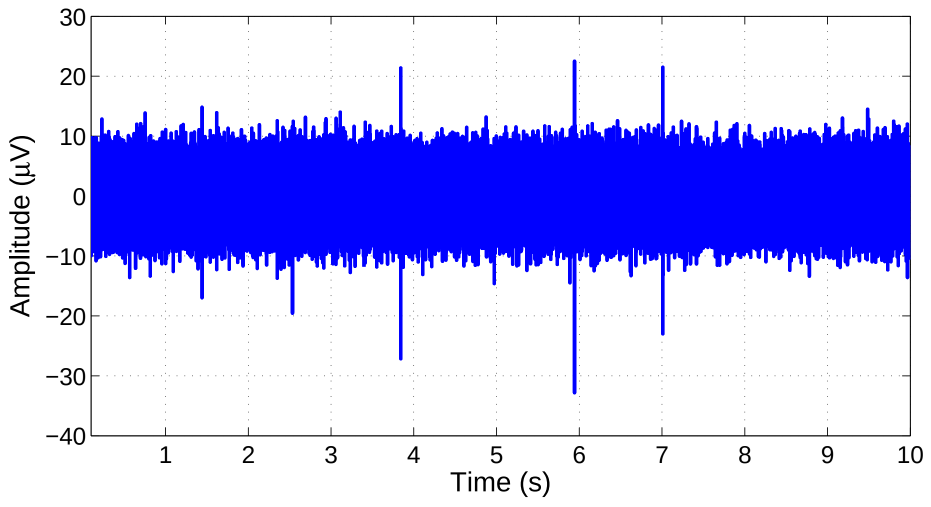
(d) Decimator output in the frequency domain

Figure 3.14: Recording system results in high-gain configuration with an input signal of 18μV at $3KHz$.

Finally, the system has been tested with a pre-recorded neural signal acquired during previous clinical trials with rabbits. The animal was subjected to vibrations at $50Hz$ and $100Hz$ in cutaneous afferents for 10 seconds. The results show how the system is capable of rejecting the huge low noise components visible in the input signal (Fig. 3.15(a)) and to highlight the neural spikes.



(a) Input Signal



(b) Output Signal

Figure 3.15: Pre-recorded neural signal processed by the recording module

3.4.2 Stimulation module tests

The stimulation module has been tested using the system presented in section 3.3.2 that permits to generate bi-phasic pulse trains with variable current, pulse width and period. A 10KΩ resistance connected between the input pin and the reference voltage was used to emulate the impedance introduced by the target neural electrodes. Fig. 3.16 shows the possibility to vary the current amplitude which, considering this output impedance, can range from 20μA to 100 μA. Higher currents can be achieved with lower impedances.

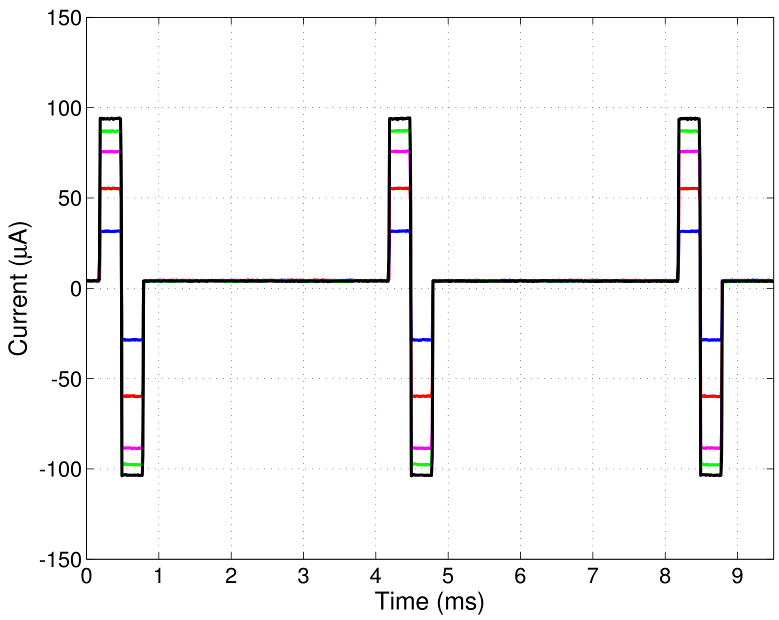
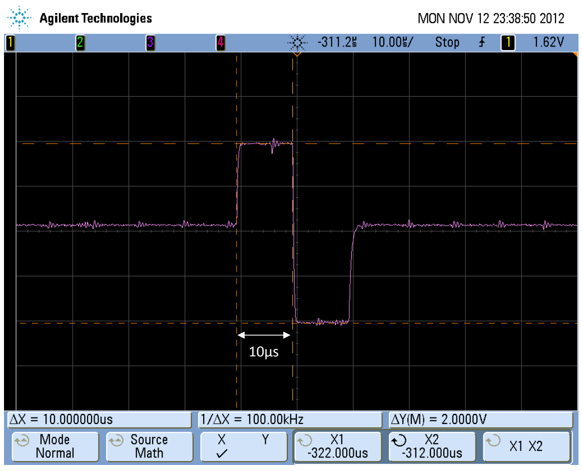
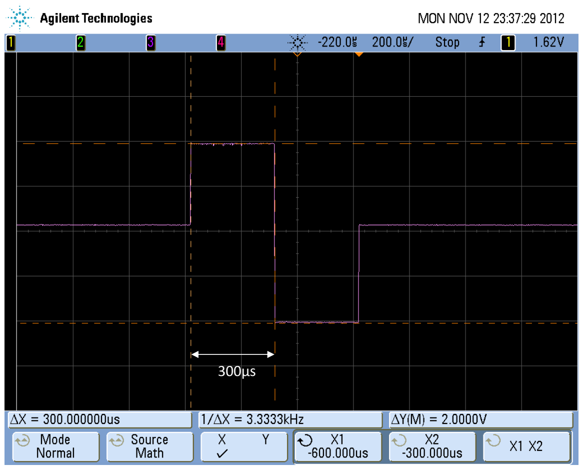


Figure 3.16: Stimulation current amplitude programmability.

Fig. 3.17(a) and Fig. 3.17(b) show how it is possible to change the pulse width W, in fact it can be programmed in a range from $10\mu s$ to $300\mu s$. The signals depicted in the figures have been acquired with the oscilloscope and they therefore represent the voltage signal drops across the equivalent input resistance connected at the electrode terminals. To obtain the corresponding current, the signals should be divided for the 10KΩ resistance value.



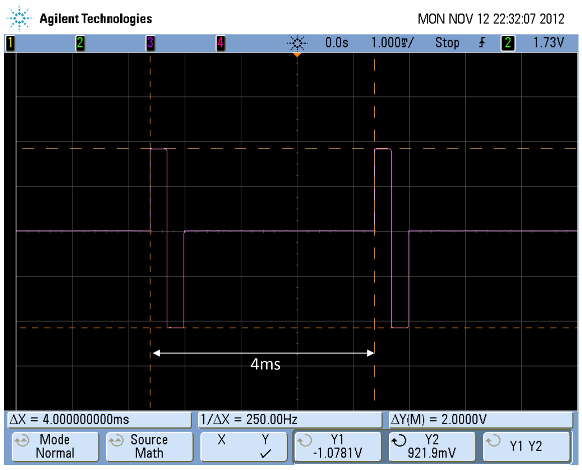
(a) pulse width W=$10\mu s$



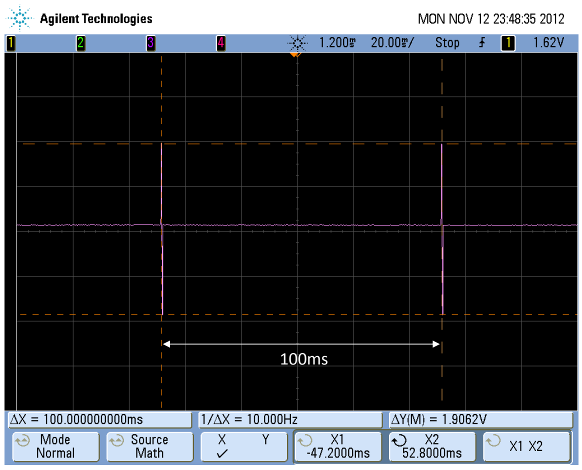
(b) pulse width W=$300\mu s$

Figure 3.17: Stimulation biphasic pulses varying the relative phase width.

The possibility to change the bi-phasic pulse period T has also been provided, it can span from a minimum of 4ms to a maximum of 100ms. Fig. 3.18(a) and Fig. 3.18(b) confirm the proper functionality of the stimulator also in this case.



(a) train period D=4ms

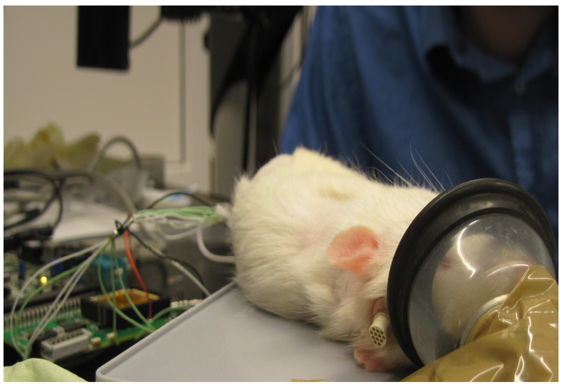


(b) train period D=100ms

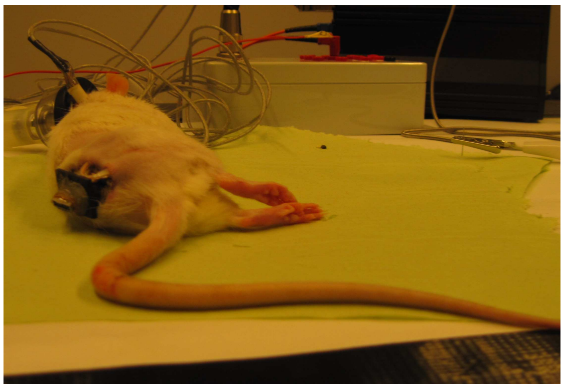
Figure 3.18: Stimulation biphasic pulses varying the relative period.

3.4.3 Real in-vivo tests

The in-vivo measurements have been performed on sedated rats at the *Ecole Polytechnique Federale de Lausanne* (EPFL) (Switzerland). An eight-channels TIME (Transverse Intra-fascicular Multi-channel Electrode) was chronically implanted in the sciatic nerve of the animal. All processes were performed using a protocol approved by the local Ethical Committee. The tests were performed after a month from the electrode implantation, therefore the results should be considered highly representative of what can be obtained in a long-term implant, when the electrode-tissue interface is already degraded. In Fig.3.20 two pictures of the experimental set-up with the chip connected to the neural electrode chronically implanted in the rat is shown.



(a) Experimental Set-up



(b) Chronic TIME implantation

Figure 3.19: In-vivo tests at the EPFL laboratories.

During the tests the hind pow of the animal has been subjected to flexo-extensor movements. The results, concerning seven different channels, have been reported in Fig.3.20.

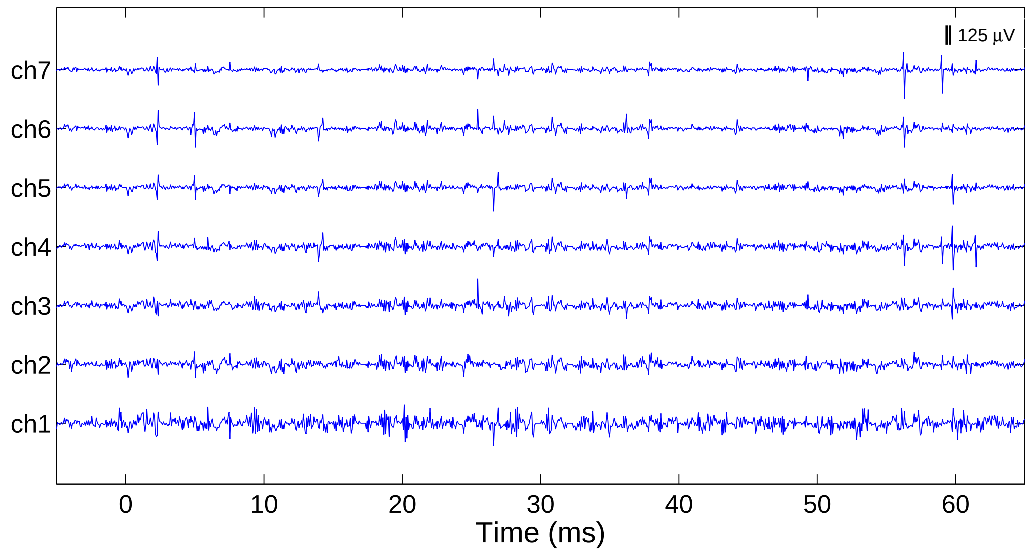


Figure 3.20: In-vivo recording results with seven channels

The presence of neural spikes with amplitudes of few tens of microvolts is evident as well as the correlation between the near channels. The test successfully confirms the system capabilities of recording neural signals with an input referred noise of less than $10\mu V_{pp}$. Fig. 3.21 shows a single spike recorded with the proposed system. As expected for a neural signal, it is characterized by an amplitude of about one hundred of μV and a duration of $300ns$.

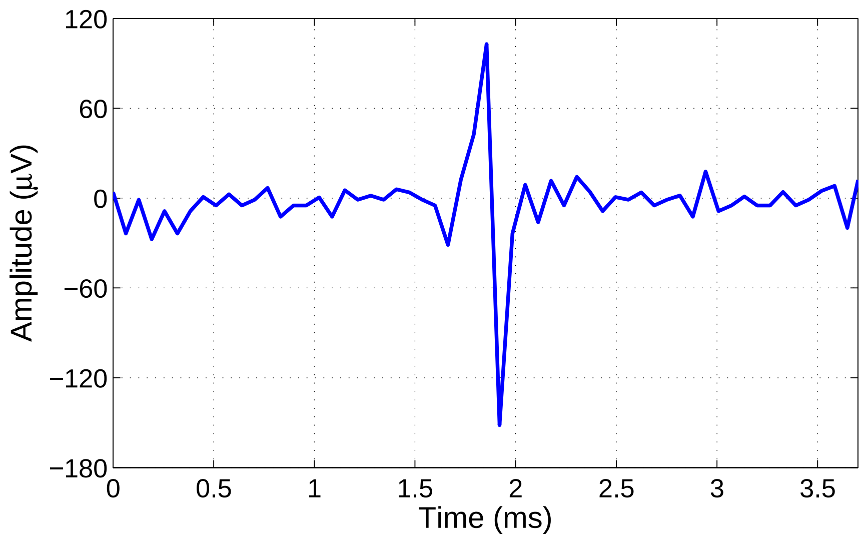


Figure 3.21: In-vivo recording results: zoom on a single spike

Chapter 4

Real-time neural signals decoding onto off-the-shelf DSP processors

4.1 Introduction

Considering the reference architecture of the closed-loop system used for the upper limb neuroprosthesis, at downstream of the multi-channel sigma-delta decimator, the high-resolution outputs at the Nyquist sampling frequency must be analysed in order to determine the desired intentions of movements of the amputee. For this reason, it will be presented the contribution related to the decoding algorithms of peripheral neural signals and the development of the correspondent hardware implementations in order to fulfill the tight timing and power constraints imposed by the reference application to allow portability of the resultant solution, in this first case targeted on a commercial off-the-shelf DSP processor.

As it has been analysed before, several researches have focused their efforts on the identification of algorithms able to decode the mechanisms with which the brain generates the electrical stimulations responsible of transmitting information related to the desired motor actions from the central nervous system to peripheral one. The aim is to extract the intentions of movements of the amputee by analysing the physiological signals, acquired by the electrodes and digitalized with high resolution by the sigma-delta converter, and to generate the correspondent electro-mechanical stimulations which must be given in input to a cybernetic hand.

As said before, the ENG-based prosthesis represents the most promising solution on which this part of the research activity has concentrated its efforts. In particular, it will be analysed the development and the implementation of an ENG-based decoding solution on an embedded system, considering an off-line algorithm of the state of the art [16] as reference which has been already verified by real experimental tests on sedated animals presenting excellent performance [69]. At the moment, the fundamental aspect of the correspondent real-time hardware implementation has been often overlooked, notwithstanding the impact that limited resources available on embedded systems may have on the efficiency/effectiveness of any given algorithm. The same results must be in fact confirmed even in case of using low operating frequency and strict low-power constraints imposed by the application into consideration in order to allow portability of the resulting device.

The aim of this part of the work is to implement a system which can work autonomously and in an unsupervised way based on the current characteristics of the acquired signal, in real-time, by using compact battery-powered devices such as programmable DSPs and FPGAs. In the following sections, it will be described the target algorithm for PNS signals decoding and the relative optimizations for the full implementation onto a off-the-shelf floating-point digital signal processor (DSP) for on-line processing. Beyond low-level optimizations, different solutions will be proposed at a high level in order to find the best trade-off in terms of effectiveness/efficiency. A latency model, obtained through cycle-accurate profiling of the different code sections, will be presented in detail in order to perform a fair performance assessment and to verify the correct functionalities even in case of worst case timing conditions.

4.2 A state-of-art ENG-based signal processing algorithm

The majority of the decoding algorithms adopted for prosthetic applications involve the generation of a mathematical model created by examples with which it is possible to understand the desired amputee movement by information related for instance to the spikeness of the active neurons in a defined time window. For this reason, it is fundamental to have electrodes with a high degree of selectivity in order not to limit the results at classification downstream. During the years, among the various proposed approaches able to acquire the ENG signals at PNS level, neural interfaces based on the use of thin-film Longitudinal Intra-Fascicular Electrodes (tf-LIFEs) have already demonstrated the capability of extracting the significant activity of each neuronal cell [59]. They are placed in parallel way to the axe of the residual nerve accessible at the stump. The relative geometric structure is shown in Fig.4.1 which highlights the presence of more sensitive sites in order to allow a multi-channel acquisition and processing.

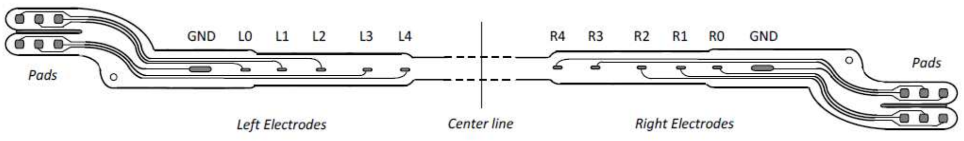


Figure 4.1: Structure of the tf-LIFEs.

The same interfaces can be used in a closed-loop system as a bi-directional communication link between the brain and the prosthesis. As said before, it involves a recording phase along the direct path from the CNS to the PNS and a stimulation phase in the opposite direction to restore the external sensory feedback to an upper limb amputee. The original work, considered as the starting point for this part of the research, has been presented in [16] with the aim of verifying if the proposed off-line algorithm was able to decode the information related to external stimuli applied on 6 adult sedated rabbits from the signals extracted by the tf-LIFEs implanted in their sciatic nerve.

This algorithm consists of 4 successive processing stages: wavelet denoising (WD), spike detection (SD), spike sorting by template matching (SS) for feature extraction and finally, classification (CL). In comparison to other state-of-art algorithms, this one presents a feature

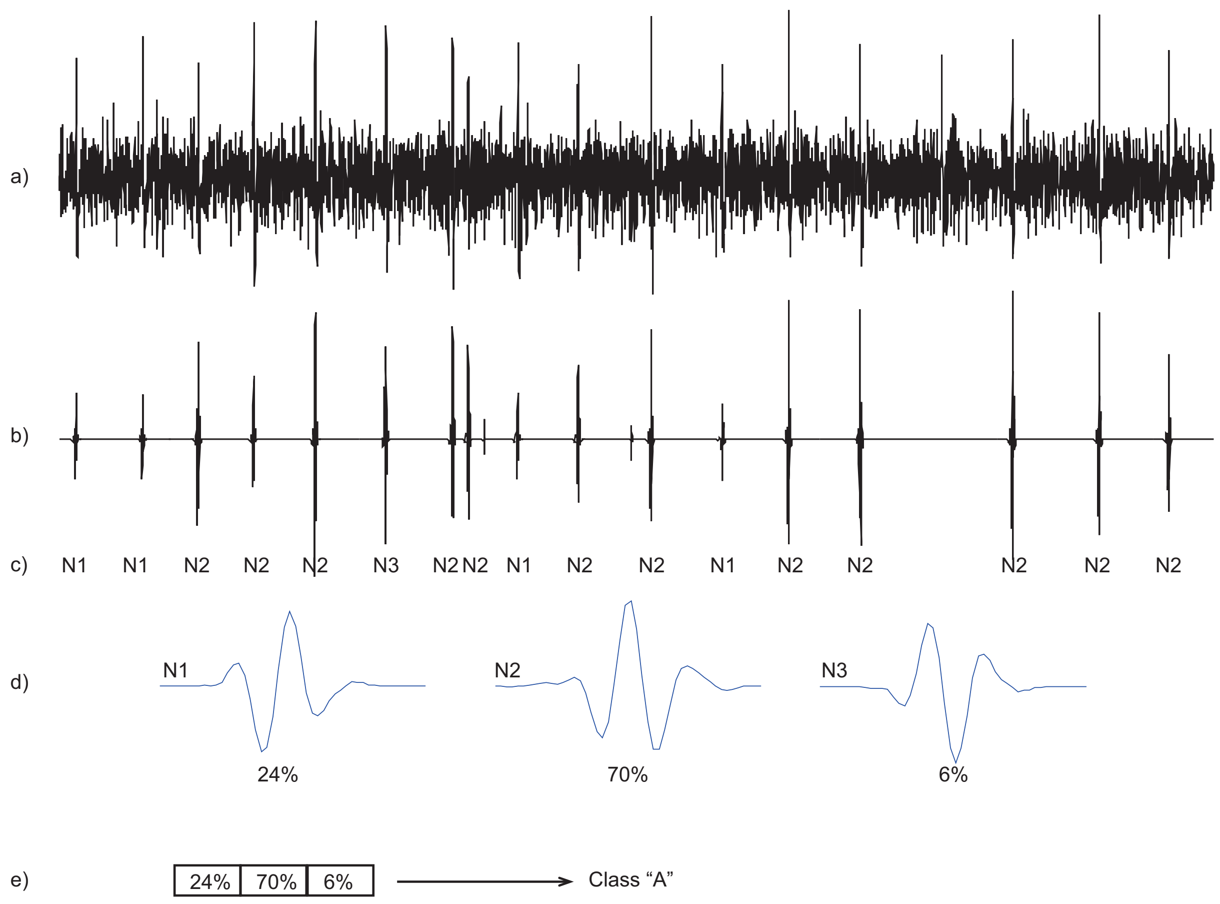


Figure 4.2: Schematic representation of the PNS decoding process on a real signal.

extraction phase, to feed the classifier of the final stage, based on typical signal processing techniques which can be easily adapted for an efficient real-time implementation on DSP architectures, as it will be detailed. At the same time, the template-matching approach is critical in terms of computational complexity, inserting difficulties on the fulfilment of the timing constraints of the application under consideration.

For these reasons, a block-on-line approach has been adopted, applying as input a stream with sampling frequency f_s of $12kHz$ and obtaining as output a class assigned by the classifier to a feature vector, called *pattern*, at a rate f_p of $4Hz$. An additional trigger with the same f_s has been processed to provide an on/off flag giving information that the underlying neural activity extracted by the tf-LIFEs is consequent to the external stimuli applied to the sedated animals and that it must be used for decoding or for the training phase of the classifier. A pattern is a vector of n_{feat} features which each of them represents the percentage of spikes in a timing window of L samples with a morphology similar to one of the ordered n_{feat} templates created in the training phase of the SS stage. A simple representation of the processing of the neural signals is presented in Fig.4.2.

In particular, the raw signal (a) recorded by the acquisition system is filtered by the WD approach (b), rejecting the samples which correspond to the noise and unchanging those of the significant spikes which are identified by means of a SD stage and compared to the previously extracted templates created by cross-correlation measures. The occurrences of the spikes with similar morphology with respect to these templates are then determined (c) as a percentage of the total pulses identified in a fixed time window. These values represent the components of a feature vector (d) which must be given in input to the classifier in order to determine the resulting class related to a particular sensory stimulus.

To achieve a good spikeness for the pattern, intended as the number of spikes per window, L should be large enough to contain several spikes and should slide on the input signal with a frame rate r adequate to follow local variations in the signal in real-time. Given t_L, the frame rate r defines the time instants t_n in which the patterns will be extracted to be processed by the classifier, i.e. $t_n = n\frac{t_L}{r}$ with $n = 1, 2, \cdots$. It is possible to choose $L = b\_len \times r$, so that b_len is the number of new signal samples, forming a $block$, needed to push on the window (the window slides with an overlap of $1 - r^{-1}$, e.g. 75% for $r = 4$). Since only the classifier requires to operate on the whole window, a “virtual” sliding window is implemented in the code for the first three stages, the last one being in charge of preserving a memory of the spike sorting results for the latest r blocks (updated block-wise) to compute the features for a single pattern. This choice reduces the memory requirements and the computational redundancies.

To provide an overview of the proposed algorithm in its on-line version, Algorithm 1 describes with a pseudo-code the different processing parts executed every time a new block of b_len input samples is available. The different conditions responsible of the transitions between the states of the algorithm can be automatically set by the code at run-time (e.g. whenever an established number of spikes has been processed, after a specific amount of time, in response to an external trigger).

The sequence of the macro-states of the algorithm changes over time in order to achieve the automatic tuning of the various stages before the normal operation:

1. WD with threshold tuning in loop with SD;
2. WD and SD with established thresholds, followed by SS in Tuning Phase (TPh);
3. templates reduction and definition of the final set of n_{feat} templates;
4. WD and SD with established thresholds, followed by SS in Steady Phase (SPh);
5. CL creation and training;
6. WD and SD with established thresholds, followed by SS in SPh and then by CL (pattern recognition).

In the following, the different stages are analysed considering optimizations of both the original algorithm, to improve efficiency and effectiveness in the context of an embedded system implementation.

4.2.1 Wavelet Denoising

WD is a preliminary processing step used in several biomedical applications to remove the background noise added to the signal of interest, in particular when it can be modelled as a Gaussian distributed random source. It is a non-linear filtering process aimed at the rejecting of the in-band and the out-of-band samples which belong to these noise sources. Such a technique transforms the raw data into an orthogonal time-frequency domain by decomposing the Nyquist bandwidth in sub-bands of different resolutions (*analysis* phase), then

Algorithm 1 Main loop of the on-line algorithm, for every new block of data (b_len samples)

```
//WD
Wavelet decomposition ($N_{scales}$ scales)
if $1^{st}$ block then
    Evaluate the parameters to compute the WD threshold, for this block
    Assign them to all the $r_\sigma$ elements required to compute the WD threshold for a whole window
    Compute the WD thresholds at the different scales
else if $tuning\ threshold$ then
    Evaluate the parameters to compute the threshold, for this block
end if
for $i = 1$ to $N_{scales}$ do
    Perform thresholding at scale $2^i$
end for
Wavelet recomposition
//SD
Spike detection (different approaches)
//SS
Crosscorrelate incoming spike with existing templates (if any)
if $template\ creation$ mode then
    if $max(correlation) > th_c$ then
        Update corresponding template by synchronized averaging
    else
        Create a new template from incoming spike
    end if
else if passing from $template\ creation$ mode to $sorting$ mode then
    Perform templates merge and reduction to $n_{feat}$ templates
else if $sorting$ mode then
    if $max(correlation) > th_c$ then
        Periodically update corresponding template by synchronized averaging
        Update templates count for feature creation of current pattern
    end if
end if
if $low\ spikeness$ AND $tuning\ threshold$ then
    Compute the thresholds at the different scales
    Update the WD threshold buffers
end if
//CL
if $sorting$ mode then
    if $training\ classifier$ then
        Classifier training
    else
        Classify the current pattern
    end if
end if
```

applies *thresholding* on the resulting coefficients introducing the non-linearity and finally transforms back into the original domain by recomposition (*synthesis* phase).

Since the wavelet decomposition represents a matched filtering between the signal and scaled-shifted versions of a mother wavelet as in the Eq.4.1 where a is the scaling factor while b represents the shifting one, the latter must be chosen with a shape similar to a typical action potential in order to emphasize the presence of the pulses. For example, in the original algorithm, it has been used the *Symlet 7* family or otherwise it can be designed according to the characteristics of the signal under consideration.

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{4.1}$$

At each stage of decomposition, WD is usually implemented with a pair of Quadrature Mirror Finite Impulse Response (FIR) filters implementing respectively a low-pass $H(z)$ and a high-pass $G(z)$ filter. They equally split the bandwidth of the input signal in approximation $a_{2^i}(z)$ given in input to the next stage 2^{i+1} and detail $d_{2^i}(z)$ which is passed through thresholding. Considering an input sampling frequency f_s, the former is composed by the components until $f_s/4$ whereas the latter with components between $f_s/4$ and $f_s/2$. In case of orthogonal wavelets, as it happens for the majority of the families, the coefficients of the filters $H'(z)$ and $G'(z)$ at recomposition are simply the mirrored versions of the relative filters at decomposition.

To achieve timing shift invariance of the WD process, it has been considered the *Ã -trous* solution [80] which is the best one for a real-time implementation thanks to its limited complexity in comparison to others, such as those based on the cycle spinning as in the original off-line work [16]. It determines that the coefficients of the analysis filters at the 2^i stage are simply obtained by oversampling applied to the coefficients of the filters at the previous stage, always preserving the same sampling rate. Such an approach allows to substitute, at decomposition, the various filtering paths of the trellis with the equivalent filters, minimizing the code complexity and the memory requirements for the mapping of the application into a DSP architecture. At recomposition phase, it is fundamental to consider a 0.5 factor for the coefficients of the synthesis filters in order to properly reconstruct the output denoised signal. A block diagram of the filters trellis used for the WD is shown in Fig.4.3.

In the original algorithm, WD works at $@f_s$ equal to $12kHz$ on the raw data performing 3 levels of decomposition/recomposition with a *Symlet 7* mother wavelet. Moreover, it removes the approximation signal at output of the last decomposition stage in order to obtain a resulting high-pass bandwidth between $750Hz$ and $6kHz$, rejecting the low-frequency interferences such as the EMG signals generated by the muscles near to the electrodes and overlapped to the signal of interest in terms of significant frequency bandwidth. Due to the support length of the *Symlet 7*, its use determines the excessive increase of the memory requirements proportional to the number of trellis stages in comparison to the *Haar* family. Considering other works in the literature [72], it has been decided to narrow the bandwidth adding one level of decomposition/recomposition and cleaning the first detail signal in order to obtain a resulting frequency response in the range of [375 - 3000] Hz. The augmented number of levels would introduce a latency of 196 samples with the *Symlet 7* in comparison to only 16 samples for the simpler *Haar*, which is then the best design choice in terms of efficiency for the proposed DSP implementation.

Between analysis and synthesis, it is fundamental to apply a certain delay to the detail samples in order to obtain a correct alignment in time and thus to avoid synchronization

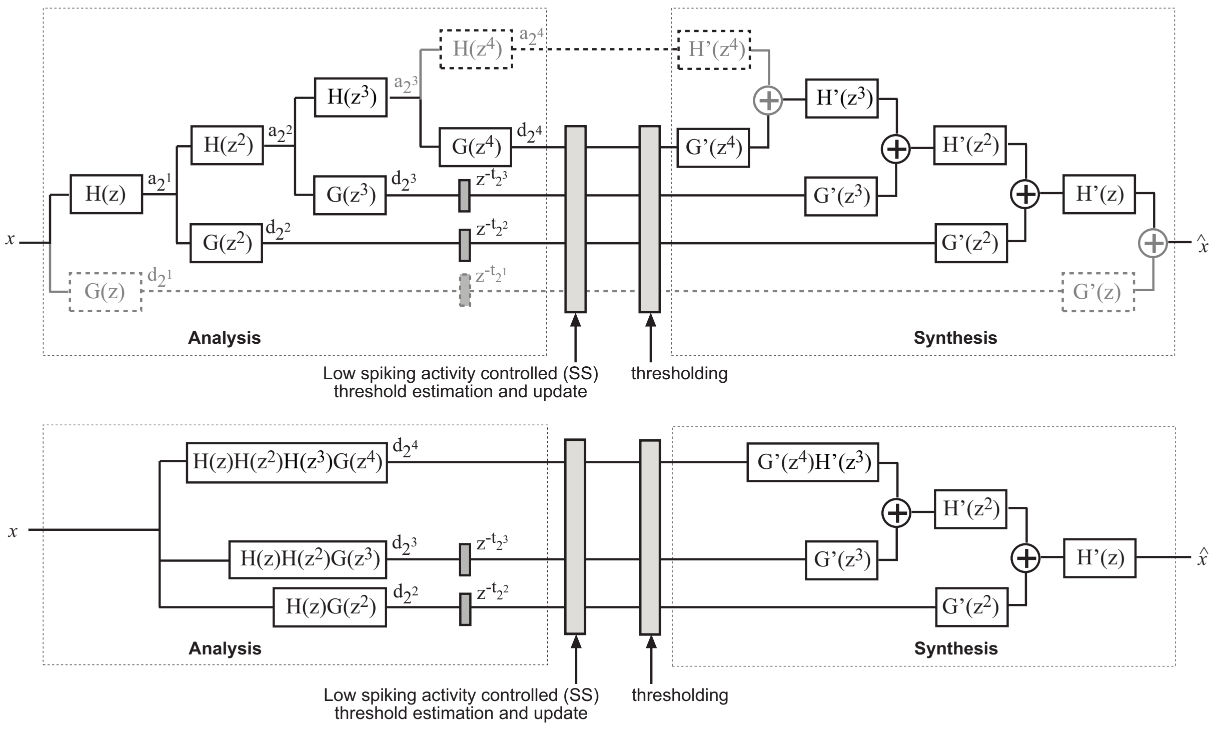


Figure 4.3: From top to bottom, the proposed approach to WD: original filter bank trellis and memory-optimized version with equivalent filters.

problems, properly reconstructing the spikes at WD downstream. This can be implemented taking into account the correct offset in the recomposition filters buffer, needing in such a way more memory locations, considering the time lag as $t_{2^i} = t_{2^{i+1}} + lat_{2^{i+1}}$, where:

$$lat_{2^i} = N_{G_i(z)} - (N_{G_i(z)}\%2) \tag{4.2}$$

$N_{G_i(z)}$ being the length of the $G_i(z)$ filter and % the reminder of the integer division.

Regarding to the thresholding phase, with the aim of obtaining the best optimization of the original algorithm, it has been preferred the *Hard* approach for its simplicity (Eq.4.3a, the detail coefficients with values lower than a defined threshold θ are rejected whereas the remaining ones passed unchanged) with respect to the *Soft* approach (Eq.4.3b, also requiring the shrinking of the input coefficients towards zero of θ value) [19].

$$Thr_\theta{}^H(x_k) = \begin{cases} x_k, & \text{if } x_k \geq \theta \\ 0, & \text{if } x_k < \theta \end{cases} \tag{4.3a}$$

$$Thr_\theta{}^S(x_k) = \begin{cases} x_k - \theta, & \text{if } x_k > \theta \\ x_k + \theta, & \text{if } x_k < -\theta \\ 0, & \text{if } |x_k| \leq \theta \end{cases} \tag{4.3b}$$

For the same reason, it is important to adopt a method for the on-line calculation of θ without inserting unnecessary complexity. To provide adaptiveness according to the characteristics of the input signal at the varying of the SNR conditions, this can be done iteratively computing θ as a scaled version of the standard deviation of the noise σ_n. The scaling coefficient can be found empirically or exploiting some approaches in literature as the minimax and the universal scaling factors. Such methods present a dependency from the length N of the signal to analyse, which is misleading in an on-line processing where the window length is extremely short. For this reason it has been used a fixed precomputed scaling factor 3.9 obtained with the minimax approach over a signal frame of 43 seconds as in [69].

The process of updating the threshold value can be performed only at the beginning and then disabled if the detected spikeness degree in the successive samples window reaches a certain level. Only when no spikes are detected by the next processing stage, the updating is enabled again. It means that the sample standard deviation can be used more efficiently and with similar results compared to the median absolute deviation typically used for its robustness to the outliers as demonstrated in [19]. It differs from what it has been done in [69] due that in this case the median absolute deviation is more appropriate for an off-line processing which needs the onerous operating of the sort of a samples buffer.

In order to evaluate σ_n over a frame of samples larger than b_len, i.e. $b\_len \times r_\sigma$, optimizing the latency, it is possible to compute for every incoming block of new b_len samples (at each scale i) their squared sum (all the detail coefficients are zero-mean) $s_{j,2^i} = \sum_{n=1}^{b\_len} d_{2^i}^2[n]$, so that for every new block of samples the last r_σ partial squared sums can be summed up, multiplied by $(b\_len \times r_\sigma - 1)^{-1}$ and then the squared root computed:

$$\sigma_{n_{2^i}} = \sqrt{\frac{1}{b\_len \times r_\sigma - 1} \sum_{j=1}^{r_\sigma} s_{j,2^i}} \tag{4.4}$$

It should be noted that r_σ is different from r and $s_{j,2^i}$ could belong to non contiguous blocks of the signal.

4.2.2 Spike Detection

Every new block of samples must be scanned sample-wise in order to find the next spike. The use of a simple absolute threshold obtained starting from the samples values at WD output has already proven its capability to be effective to detect the onset of a neuronal pulse even in case of critical real conditions [69]. Such a threshold can be either fixed or self-tunable exploiting the root mean squared *rms* of the denoised signal, the latter leading to a worse estimate when there are bursts of fused spikes, also being computationally expensive.

Such a simple approach for the SD can be improved using a further processing step based on the use of the Non-linear Energy Operator (NEO) [36] applied to the WD output samples. As demonstrated in [23], it represents the best compromise choice from the hardware point of view of the correspondent implementation even maintaining good performance in accuracy. In fact, tests performed on a publicly available dataset of synthetic signals [72] have showed that the NEO is generally less sensitive to the choice of the amplitude threshold of the SD stage, determines high values of accuracy at the cost of a reduced computational complexity and it is not very influenced by the variation of the SNR of the signal. Such benefits in detection only increase when the input signal of the NEO has been previously filtered for noise reduction. It can be used to obtain a pre-emphasis of the spikes, and then a threshold can be applied in order to select the correct time frame around the spike. The NEO is defined as follows:

$$\eta\{\hat{x}[n]\} = \hat{x}^2[n] - \hat{x}[n+1] \cdot \hat{x}[n-1] \tag{4.5}$$

It gives higher values when the signal is characterized by high power and high frequency components, which happens for the action potentials under consideration. The spike detection threshold is then obtained as a scaled version of the mean value of the NEO in a predefined time interval:

$$Thr = C \cdot \frac{1}{N} \sum_{n=1}^{N} \eta\{\hat{x}[n]\} \tag{4.6}$$

where C is chosen by empirical considerations.

Only the detected spikes whose support is at the same time shorter than a parameter len (expressing a time value in terms of number of samples) and larger than 3 samples are analysed. len includes a head and a tail of 5 samples each, which must be below the threshold, respectively before the onset and after the end. During the TPh, larger spikes are discarded but a counter is incremented in order to have a *spikeness* index for that window. During SPh, if a waveform is becoming wider that len, the end (if present) is identified as the rightmost sample below the threshold preceded by a sample above the threshold, avoiding to look for the adjacent 5 samples. This is done in order to recognize spikes embedded into bursts, even if the proposed solution does not address the problem of the fused spikes.

4.2.3 Spike Sorting

The SS is a correlation-based algorithm which must be able to work in real-time both when all the templates are being created (the *tuning* phase, TPh) and when they are used to verify their morphology similarity with the detected spikes (the *steady* phase, SPh). For the latter, the aim is to obtain the neural activity of each neuronal cell by cross-correlation measures between the current spike and the templates before created during the TPh, starting from the hypothesis that each active neuron fire pulses with a defined morphology. The Fig. 4.4 shows the flow diagram of this processing step.

Once a possible spike has been isolated by the SD, the SS is skipped in case of WD thresholds calculation otherwise the relative samples are stored in a temporary buffer of length $2 \times len$, centering the spike with respect to its maximum value. During the TPh, the algorithm can create up to M_{N_t} waveforms in a templates matrix, each one with length of *len* elements. M_{N_t} should be larger than the number of possible neurons whose activity can be detected on a single channel because populating this matrix is a blind procedure not involving a morphological analysis able to discriminate between spikes and noise. Fortunately, the number of the active cells at PNS level is quite low compared to the CNS case, limiting the memory requirements for the DSP implementations in this phase.

At the beginning during the TPh, if the current number of created templates N_t is zero, the central part of the temporary buffer, *len* samples wide, is standardized and stored in the first location of the template matrix, incrementing the N_t counter. The standardization is due to the fact that it is adopted a normalized cross-correlation method for which both the detected spikes and the templates must have a Gaussian distribution with zero mean and unit variance, easing the calculation of the Pearson's product-moment correlation coefficients. If $N_t > 0$, the algorithm enters in its main loop in Fig. 4.4 where the cross-correlations between the temporary buffer and all the current N_t templates are computed, determining a similarity value of the current standardized spike and the waveforms previously stored in the relative matrix. The maximum value of the cross-correlations M_c is compared against a threshold th_c.

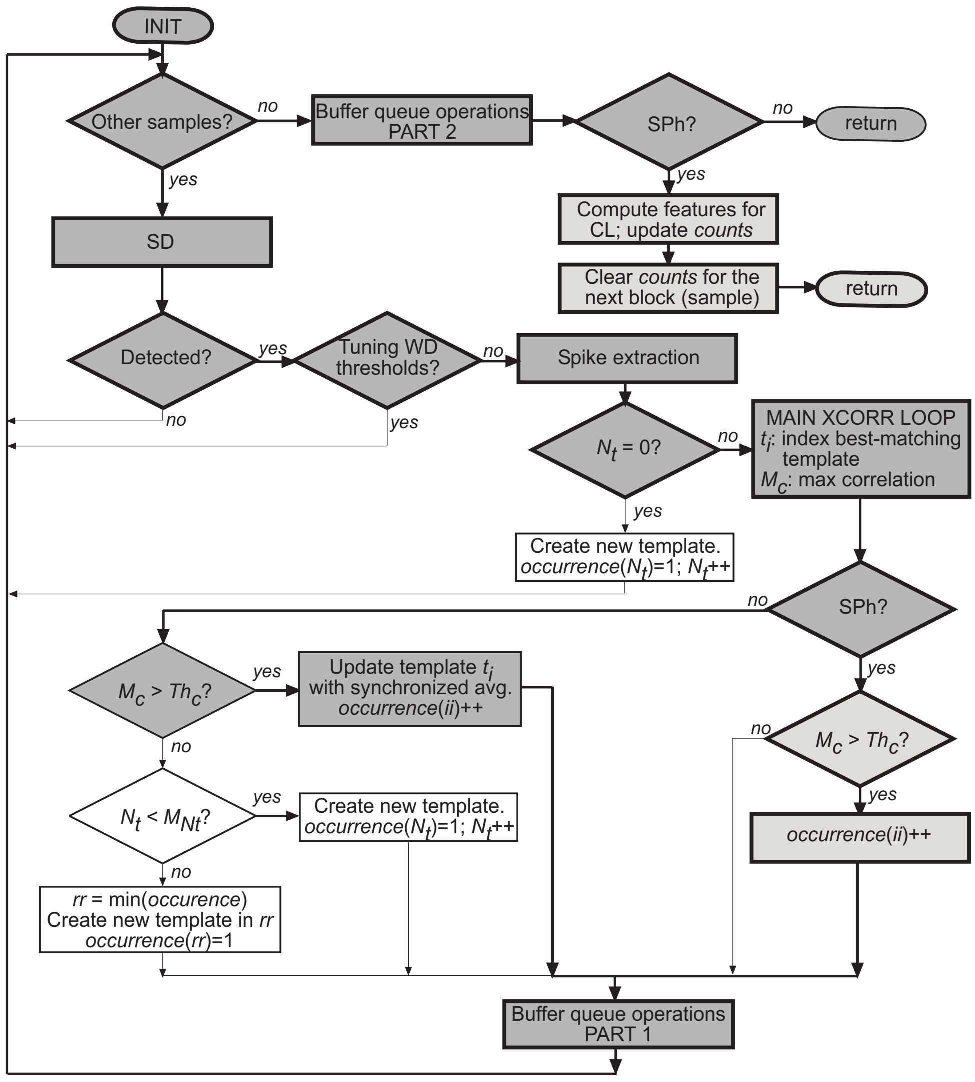

INIT
Other samples?
no
Buffer queue operations PART 2
SPh?
no
return
yes
Compute features for CL; update counts
Clear counts for the next block (sample)
return
yes
SD
Detected?
yes
Tuning WD thresholds?
no
Spike extraction
no
yes
Nt = 0?
no
MAIN XCORR LOOP
ti: index best-matching template
Mc: max correlation
yes
Create new template. occurrence(Nt)=1; Nt++
no
SPh?
Mc > Thc?
yes
Update template ti with synchronized avg. occurrence(ii)++
yes
no
Mc > Thc?
no
yes
Nt < MNt?
yes
Create new template. occurrence(Nt)=1; Nt++
occurrence(ii)++
no
rr = min(occurence) Create new template in rr occurrence(rr)=1
Buffer queue operations PART 1

Figure 4.4: The flow chart of the spike detection and the sorting steps.

During the TPh, a correlation greater than th_c leads to the update of the best matching template t_i by synchronized averaging, after the alignment of the two waveforms on the cross-correlation maximum, taking into account the number of the previous spikes with respect to which the considered template has been obtained. On the contrary, during SPh only a counter of the occurrences of the template t_i on the new block is incremented. In this phase it is also possible, by setting a parameter, to allow the template update (continuously or every N occurrences) in order to enable a continuous adaptation to small changes in the spikes morphology. After that, the temporary buffer is cleared to be ready for the next spike.

In this phase several templates are created every time $M_c < th_c$ and the templates matrix is not full. Since the templates matrix size has an influence both on the memory occupation and on the processing time, the number M_{N_t} cannot be too large. To emulate an infinite template matrix, during TPh if a new template needs to be created and $N_t = M_{N_t}$, the template with the lowest occurrence is overwritten and its number of occurrences reset. At the end of this phase, the available N_t templates are compared evaluating the cross-correlations with each other: when the maximum of a cross-correlation is above th_c the two templates undergo a weighted synchronized averaging and the process of reduction restarts from the very first template until, on a full sweep, no averaging is performed. Then only the most used $n_{feat} < M_{N_t}$ templates are retained for the SPh.

During SPh, the SS acts as a feature extractor for the classifier which operates downstream $@f_p = 4Hz$. This means that a new pattern is generated every 0.25s, and takes into account the ENG activity over $L = b\_len \times r$ samples. For such a reason, a matrix has been used to store row by row the occurrences of each template in each of the r blocks composing an $L-$wide sliding window. Such a matrix is updated column-wise at every new block and a pattern for the classifier is a feature vector obtained summing up by columns the occurrences of each template.

4.2.4 Classification

When a supervised classifier is used, labelling of the training set patterns is a necessary step in order to create a model learned by examples with which then categorize each input pattern. A trigger signal that is always zero except when the significant neural activity is present can be useful in this step, giving information about the active state during and after the moment which a particular external stimuli is applied to the sedated animal on test. Labels can be assigned automatically if it is clear the meaning of the signal under trigger (e.g. different stimuli in afferent recording or specific movement intentions in efferent ones). In an interactive training scenario, labelling can be performed exploiting GPIO pins of the processor, by manually selecting the performed (or desired) movement among a given set. The trigger signal is processed in real-time producing a trigger flag for every pattern indicating with 1 that the trigger, for the last block of samples, was high or that it was high no more than 3 seconds before. The trigger flag is set to zero otherwise.

The final features are obtained dividing every template occurrence count for the sum of all the occurrences in that window (the *spikeness* index). It is preferred a relative spikes rate in comparison to the absolute one for the same reasons explained in [16]. The spikeness index is used to establish whether or not the pattern deserves to be processed by the classifier (either in training or test), avoiding its continuous response at run-time when low ENG activity is identified. When the trigger flag is 1, the pattern is passed to the classifier only if:

$$s \geq \mu_s + \frac{1}{2}\sigma_s \tag{4.7}$$

where μ_s and σ_s represent respectively the sample mean and standard deviation of the spikeness index s computed over the whole signal which will be used for training.

The chosen classifier is a Support Vector Machine (SVM) [10] which represents a typical supervised algorithm for which representative patterns belonging to each class are known as well as the number of the existing classes. For each of them, there are a number of examples of feature vectors that characterize them. This type of algorithm involves two successive phases: the training and the test. During the first one, starting from a dataset of patterns for which it is known the relative class, a mathematical model is created at off-line, learned by examples, so that it can be consequently used to predict at on-line the class of each incoming feature vector. During the second one, the algorithm accuracy in classification is evaluated with the remainder of the input dataset.

It is a binary classifier that, starting from a set of feature vectors belonging to the training set, determines an optimal separating hyperplane in a multidimensional feature space that maximizes the distance between each vector of the two classes and the decision boundary

(Fig.4.5). The greater this distance, the lower the generalization error of the classifier. Since these points are usually not linearly separable, it is possible to map the feature vectors by a non-linear transformation on a higher dimension space where it is possible to define a linear decision boundary. This is accomplished by means of a kernel function [79] applied to the data represented in the original space.

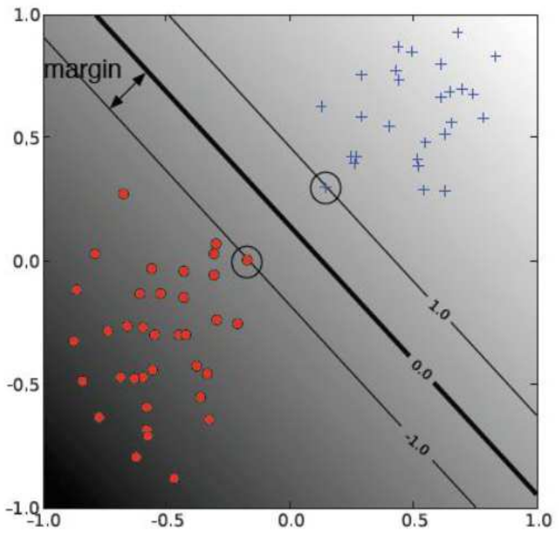


Figure 4.5: SVM hyperplane which separates the features vectors belonging to the two classes.

The SVM training is a batch processing which does not need to run in real-time because it is executed only once and the acquisition can be stopped at that time. In particular, some techniques [43] can be exploited to update the classifier model once in awhile avoiding performance degradation due to the current critical conditions for the prosthesis. This approach can be advantageous in case of several data or non-stationary data modifying the mathematical model taking into account the new patterns when they are available, without repeating the training phase using the whole dataset but starting from the current model which can be updated based on new examples.

The multi-class problem for the SVM, which was initially conceived as a binary classifier as said before, originating from the need of discriminating between several stimuli/movements has been solved by means of a "One-vs-The rest" approach [32]. Such a solution appeared to be preferable in the context of a limited-resources implementation compared to the "One-vs-One" exploited in [16]. With the chosen approach, the number of classifier models to train is equal to the number of classes q at stake whereas in case of the "One-vs-One" approach it is necessary to train $q(q-1)/2$ models, determining an excessive overhead in the comparison of the current pattern with respect to them.

In every iteration, a model is trained, i.e. the support vectors and the bias parameter of a non-linear hyperplane are obtained in order to separate the points belonging to a particular class in the multi-dimensional features space from those related to the remaining classes. After training, the classifier determines the predicted class on the basis of the minimum exponential square distance between the point in the features space and the support vectors belonging to the class related to $i-th$ iteration of the training phase.

It has been decided to use the soft-margin version of the SVM classifier (C-SVC) adopting as kernel function the Radial Basis Function (RBF) to perform the non-linear transformation. In particular, the optimal values to be assigned to the various parameters in order to maximize the performance in terms of classification accuracy and processing latency on the given datasets have been selected:

- a unitary value for the γ parameter of the RBF;
- a unitary value for the cost of the C-SVC algorithm;
- the tolerance of the termination criterion equal to 0.1.

As explained in [16], even if this previous work has exploited a different set-up, it has been decided to use fixed parameters due that their tuning can introduce unnecessary overhead for the DSP implementation. Before the training, the patterns undergo a normalization process in order to remove the bias introduced by those features having a greater range of variation than the others, maximizing the margin between the separating hyperplane and the features vectors belonging to a defined class at each iteration. Normalization leads to the features stretching in the range between -1 and $+1$. Compared to the work presented in [69], the trained classifier has been recoded. It has the role of providing in real-time, $@4Hz$ in the current version, an indication of the kind of stimuli (afferent) or the intended movement (efferent) as decoded from the ENG signal by distance measurements between the current feature vector and the separation hyperplane of the relative model. In the latter case, such an information should be used to generate the correct electrical signals to control the active prosthesis.

4.3 The DSP implementation: porting details

The problem of the implementation of advanced neural signal decoding algorithms onto embedded platforms is usually overlooked, assuming that every algorithm can be implemented in real time on such platforms. Most of the proposed algorithms are conceived starting from their off-line solutions, therefore in some cases it is impossible to develop an equivalent on-line version which allows to obtain at least the same performance. Application Specific Integrated Circuits (ASIC) and Field Programmable Gate Arrays (FPGA) can provide a very efficient implementation almost in any case, at the expenses of a complex design and very limited flexibility which must satisfy tight constraints in terms of power consumption and area occupancy according to the characteristics of the application under consideration. Microprogrammed implementations are more flexible: in this case the best choice is represented by the DSP, expressly designed for advanced signal processing. On a DSP, the limited resources represent a big challenge for the designer: the assumption above can lead to very unrealistic conclusions without an accurate investigation.

As target for the proposed hardware implementation, it has been chosen a floating-point DSP platform, the TMS320C6713 processor by Texas Instruments. This processor can run up to $300MHz$ and presents a Very Long Instruction Word (VLIW) architectural model enabling the execution of up to 8 arithmetic operations (mixed fixed and floating point) in parallel on different data. The efficiency of the implementation on this kind of processor strongly depends on the VLIW code density, representing the number of instructions that can be actually executed in parallel. The C6000 compiler is very helpful from this perspective but the best results can be obtained only by means of optimized libraries and an adequate coding technique, also respectful of the processor memory organization. The chosen DSP presents a memory hierarchy organized in three levels. There are 2 separate cache memories for data and instructions (Level 1, L1) for a total of 4kB, and a common configurable RAM/2nd level cache memory by 256kB (Level 2, L2). The L2 cache (configurable to be used as a RAM, 1-way, 2-way, 4-way cache) can be as large as 64kB (4-way). The external memory (Level 3, L3) is off-chip. It should be noted that the access to L3 introduces a penalty of several clock cycles compared to the access to L2, so memory allocation is not a secondary issue in this kind of architectures.

The application code has been developed in C programming language. All the parameters requiring an on-line tuning are automatically set up by the algorithm at run-time without any interaction with the user. The SVM training, performed off-line in Matlab in the first tentative real-time implementation of the algorithm [69], has been also integrated on the DSP. Carrying out the whole training process on the embedded system has significant reverberations on the possibility of performing the training of the prosthetic device autonomously by the patient himself without external tools. For the sake of the embedded porting, the original C++ code of the LIBSVM [13] has been deeply modified in order to transform it in plain C avoiding the object oriented parts unsupported in most DSP platforms. Only the minimal set of functions required to properly operate with the selected design choices has been ported. This led to a code with a reduced memory footprint and extremely portable.

Some optimizations, which can be enabled or not, limit the portability to the processors of the same family. For instance, data movements (e.g. buffer updates and initializations) are managed by the Enhanced Direct Memory Access (EDMA) peripheral in order to execute them in parallel with the processing, with no cost in terms of CPU cycles except for the transfer initializations. Advanced DSP library functions can be enabled at compile time with the aim of improving the VLIW code density. In particular, thanks to the block on-line approach, the *DSPF_sp_fir_gen()* function has been used for the WD stage for the FIR filtering, the *DSPF_sp_maxidx()* in the SS both for the identification of the maximum of every cross-correlation and for the identification of the best matching template, the *DSPF_sp_w_vec()* has been used as a weighted sum of vectors for the synchronized averaging, *DSPF_sp_vecsum_sq()* for the computation of the standard deviation in the cross-correlation (squared sum of vectors) along with the *DSPF_sp_dotprod()* one, which performs the dot product of 2 vectors. It should be noted that the particle *sp_* in the function names above reveals a single precision floating point computation, but also the double precision is supported in the same library. Another library, the fastRTS one, has been used to speed up the processing of scalar math operations, as the reciprocal of a number (*recipsp()*) and the squared root (*rsqrtsp()*), again in single precision. All the advanced coding practices for this kind of platform have also been exploited and the code has been compiled with the highest optimization (-o3). A great attention has been also spent in the memory allocation in order to exploit at the most the internal memory of the DSP.

4.4 Test Data

Different datasets have been used for testing, with different purposes. In order to deeply evaluate the performance of the first three stages of the algorithm, leaving apart the CL, it has been decided to exploit a synthetic dataset at first as in [72]. It has been constructed starting from a database of 594 physiological real spike waveforms obtained during experimental recordings and extracted from neocortex and basal ganglia at CNS level. Those considered as test data for the proposed analysis are composed by pulses generated by three neurons, each one which fires action potentials with a particular morphology.

To simulate the background noise, spikes at random times and amplitudes coming from neuronal cells placed at a greater distance with respect to the point in which the electrodes are implanted, were overlapped to the relevant neural activity of the limited number of neurons. This is a realistic condition for cortical implants but not for PNS signals, suffering from other physiological (mainly EMG) interferences. Such a dataset, providing a ground truth due to its synthetic nature, can be also used for PNS-oriented algorithms with tf-LIFE electrodes because even in this case the impulsive sources are limited in number by the electrodes selectivity: in fact, intra-fascicular electrodes do not allow to isolate the neural activity of a single neuron on each channel. This implies that an active pad of the electrode picks up action potentials coming from multiple axons, characterized by different amplitude and morphology, thus requiring a properly designed SS.

The synthetic dataset includes different signals (called Easy1, Easy2, Difficult1, Difficult2) with increasing levels of complexity. The complexity is related to the similarity in the morphology of the action potentials of the three neurons. Figure 4.6 shows the correlation between couples of action potentials in the signals. The correlation values are below 0.8 only for the first dataset (Easy1), with a peak at 0.98 for Difficult2 with a noise level of 0.2. In this scenario, an algorithm based on template matching as the proposed on-line one cannot perform an accurate spike sorting but this is not the typical situation in real PNS recordings. At the same time, in the light of this assumption, in the following section the results achieved on the Easy1 dataset should be considered with the greatest attention.

In this case, the neural signal bandwidth is considered in the range between 300 Hz and 3 kHz. Data have been originally created using a sampling frequency of 96 kHz and then downsampled at the frequency of 24 kHz, considering spikes with support length of 64 samples. The background noise level is defined in terms of its standard deviation and it ranges from 0.05 to 0.20 but, only for the Easy1 dataset, it can reach 0.40. The average firing rate is 20 Hz, with a refractory period of 2ms, and the number of pulses associated to each of the three neurons is approximately equal to 30-35% of the total number of spikes in each signal.

The synthetic database does not allow the CL stage testing because they don't have information about external stimuli (afferent) or intentions of movements (efferent). To this aim, it has been exploited real afferent signals coming from the PNS to train and test the SVM classifier. These signals, kindly provided by Prof. Xavier Navarro and his team (Universitat Autònoma de Barcelona), have been acquired through Longitudinal Intra-Fascicular Electrodes (LIFE) in the sciatic nerve of a sedated rat and recorded according to the protocol used in [74]. Five classes of sensory events can be identified on segments of the available signal. More in particular, these are touch sensation elicited over the four different areas of the rat limb (A to D) stimulated with different Von Frey filaments, and a class associated to flexion movement performed with animal's hind limb. Labelling was performed exploiting a trigger signal as in [69]. The whole dataset has been randomly divided using 80% of samples

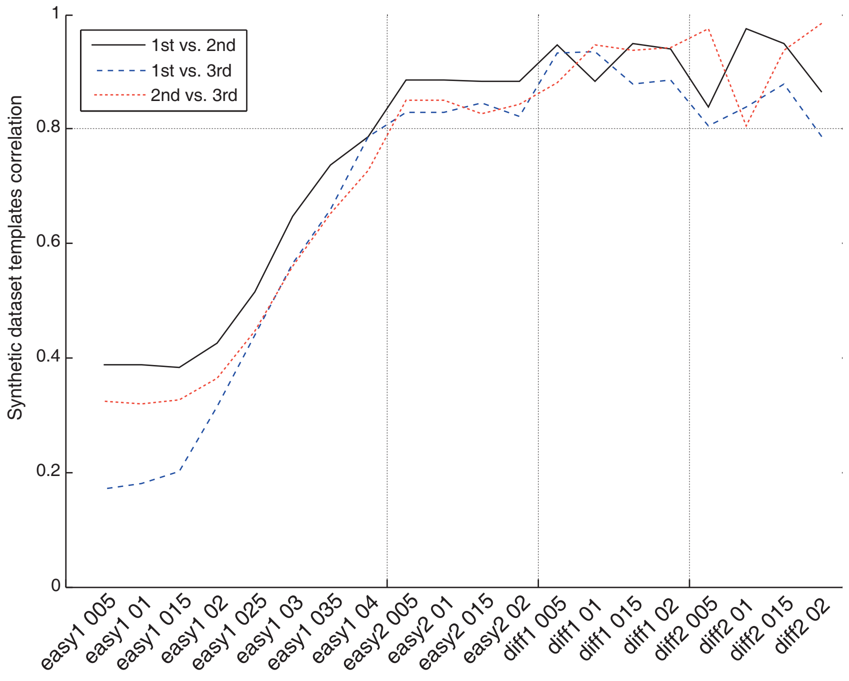


Figure 4.6: Correlation among the various couples of spike templates for each dataset.

for the training set and the rest for the test set, for several randomized trials, in order to evaluate the average accuracy of the on-line version of the reference algorithm implemented on DSP in real operating conditions.

4.5 Experimental Results

The proposed algorithm and its implementation on DSP have been analysed in terms of effectiveness (quality of the proposed solution in different scenarios) and efficiency (in the light of a real-time implementation). For a comparative evaluation of the former aspect, the on-line algorithm (except the CL stage) has been compared to a top state-of-the-art off-line algorithm for spike sorting such as the unsupervised super-paramagnetic clustering (SPC) [72] included in the *WaveClus* tool. It consists of the following processing steps:

- a 4^{th} order non-causal bidirectional elliptic IIR bandpass filtering between 300 Hz and 3 kHz;
- SD using an amplitude threshold on the filtered signal;
- SS based on super-paramagnetic clustering.

The bandpass filtering has the main role of removing the background noise. The bidirectional approach limits the distortion on the detected spikes morphology typically introduced by IIR filters, at the expenses of a limited increase in the computational complexity. The SD stage uses a positive amplitude threshold, applied on the output signal of the elliptic filter, derived from the estimation of the standard deviation of the noise as $Thr = 4\sigma_n$ where

$$\sigma_n = median\left\{\frac{|x|}{0.6745}\right\} \tag{4.8}$$

A time windowing ensures the extraction of templates limited to 64 samples at the sampling frequency of 24 kHz for each detected spike. The algorithm assumes a refractory period of about 2 msec, thus not including any strategy for the overlapped spikes. For each detected spike, the algorithm uses as features for the classifier the wavelet transform coefficients that meet the criterion of Kolmogorov-Smirnof normality (i.e. only those that have a multi-modal distribution) and able to determine the best performance in the SS. As last stage, the algorithm includes the unsupervised SPC, originally presented in [9]. Such a clustering automatically selects the temperature parameter value, exploited to influence the cluster size and modifiable by the user in order to really achieve the best performance, through a Monte Carlo simulation. Such an off-line algorithm has only the objective of deriving the spike templates associated with each neuron of interest (i.e. single-unit activity), since for the prediction of the movement intention a successive stage would be required.

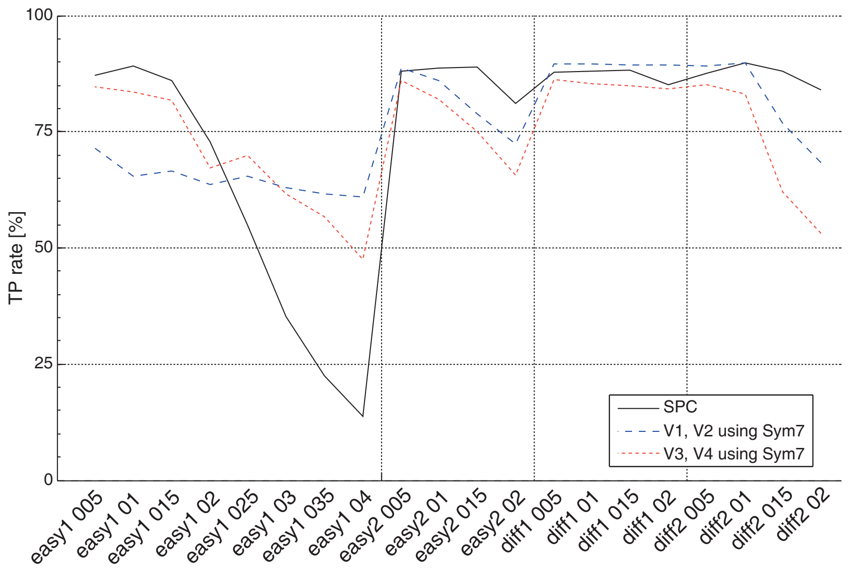
4.5.1 Effectiveness analysis

Compared to the original algorithm [16] and its first tentative embedded implementation [69], several modifications have been implemented at the level of the WD and SD stages, beyond the complete porting of the classifier and its improvement, giving rise to different versions:

- V1 is similar to [69] (WD with 3 levels, removing approximation a_{2^3}, bandwidth 750Hz-6kHz, SS on the output of the WD stage) but the NEO has been introduced in the SD;
- V2 is a modified version of V1 (WD with 3 levels, removing approximation a_{2^3}, bandwidth 750Hz-6kHz, NEO for SD) but the SS is performed on the output of a band-pass filter between 300Hz and 3kHz as in [72], using the WD stage only for the SD;
- V3 is a modified version of V2 (NEO for SD, WD only for the SD, SS on the band-pass filtered signal and not on WD output) but exploiting a WD with 4 levels, removing approximation a_{2^4} and also the detail at the first scale d_{2^1} leading to a bandwidth of 375Hz-3kHz which is more similar to that of [72];
- V4 is similar to V1 (NEO for SD, SS on the output of the WD stage) but, as in V3, the WD has been modified to 4 levels, removing approximation a_{2^4} and also the detail at the first scale d_{2^1} leading to a bandwidth of 375Hz-3kHz.

Both the minimum correlation threshold th_c and the coefficient C in the NEO have been determined experimentally in order to improve the performance of the algorithms in relation to the different signals used for testing.

At first, the different versions of the proposed on-line algorithm have been evaluated on the synthetic database in order to have a ground truth and to be able to compare them against the off-line SPC algorithm [72]. In this case, the performance has been evaluated in terms of True Positives (TP) and False Positives (FP). In particular, the TP rate has been used, representing the percentage of true detected spikes over those actually present in the



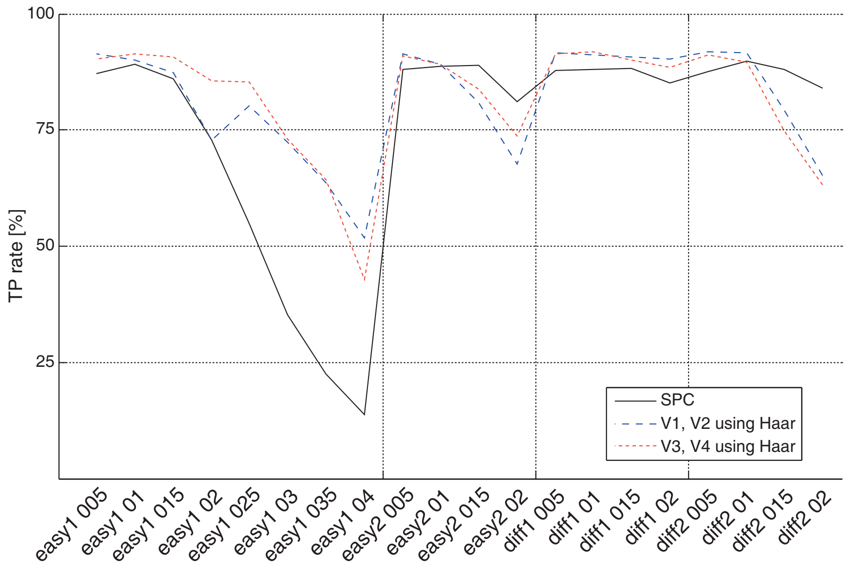


Figure 4.7: Spike detection results in terms of TP rate using the Symlet 7 (top) and the Haar (bottom) wavelets compared to SPC.

signal. Furthermore, rather than simply using the number of FP, the FP per minute have been computed.

To evaluate the usefulness of the WD on the successive SD stage, it has been tested the different versions of the algorithm with both *Symlet 7* and *Haar* wavelets looking at the TP rate (Fig. 4.7). In both cases, the proposed SD algorithm, joining WD and the NEO, is superior to the simple amplitude threshold applied to the band-pass filtered signal in the SPC for the largest part of the signals, the *Haar* wavelet revealing superior performances than the *Symlet 7*. Another interesting related metric is the number of FP per minute, related to the detected spikes belonging to the background noise. In this case, the superior detection performance could lead to a worse performance. However, Fig. 4.8 reveals that the proposed SD including a WD pre-filtering performs better than the SPC detection, with isolated exceptions on the Easy1 database. At the same time, the *Haar* wavelet seems to lead to a poorer de-

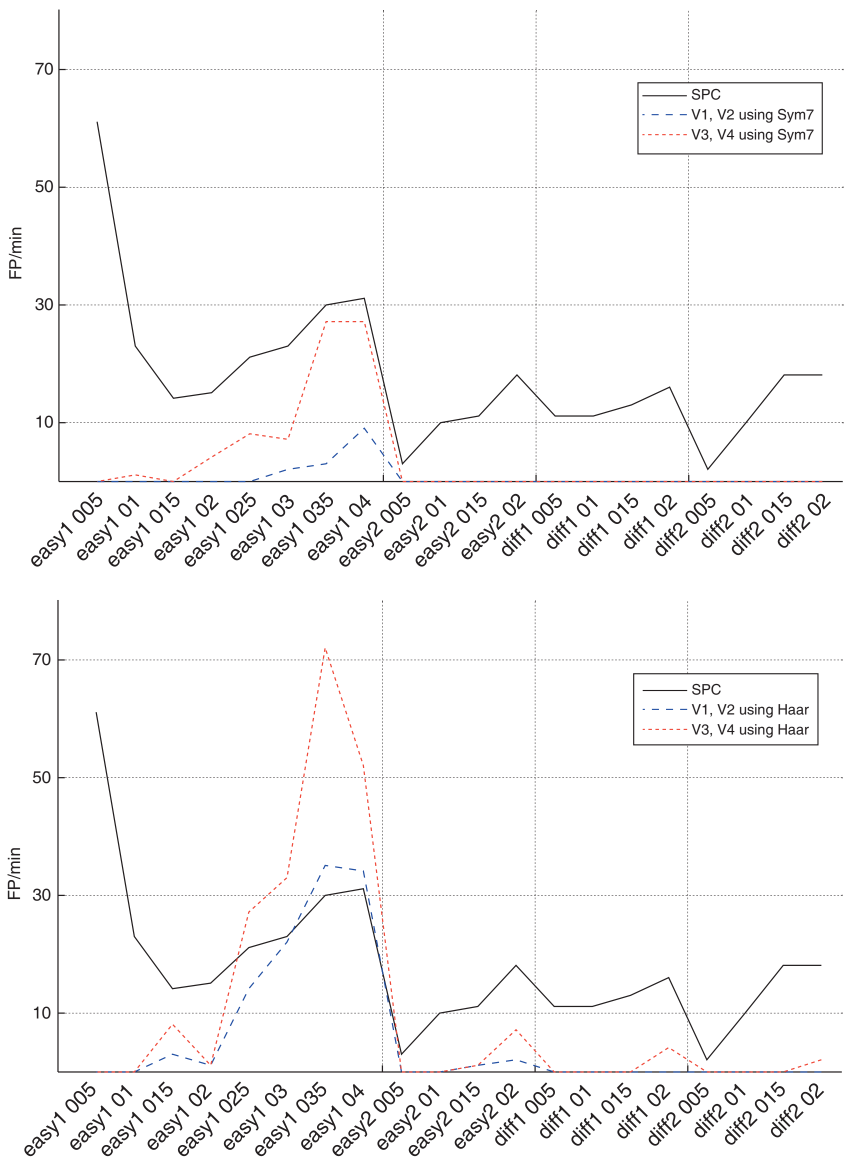


Figure 4.8: Spike detection results in terms of FP per minute using the Symlet 7 (top) and the Haar (bottom) wavelets compared to SPC.

noising and then to an increased number of FP in the signal. From the above results, it seems that the combined WD and NEO lead to an effective SD that, in case of the *Haar* wavelet, is also very efficient from a computational perspective for a real-time implementation.

In terms of percentage of matching, from Tab. 4.1 it is possible to see that the results seem to be strongly influenced by the noise level and are largely worse than those achievable off-line with the SPC for all the dataset but the Easy1 one. The problem is the correlation-based approach exploited in the on-line algorithm, unable to correctly operate on very similar spikes. In fact, the best performance can be achieved with the V4 version of the algorithm and the *Haar* wavelet, limitedly to the Easy1 dataset, with even better results than the SPC for some noise level. Results are even worse for the versions of the on-line algorithm performing the SS on a band-pass filtered version of the signal rather than on the wavelet denoised one, with increasing levels of noise. Again the *Haar* wavelet seems to be more effective than the

Table 4.1: Template matching percentage using the Haar and the Symlet 7 mother wavelet.

	Easy1_noise005	**Easy1_noise01**	**Easy1_noise015**	**Easy1_noise02**	**Easy1_noise025**	**Easy1_noise03**	**Easy1_noise035**	**Easy1_noise04**	**Easy2_noise005**	**Easy2_noise01**	**Easy2_noise015**	**Easy2_noise02**	**Diff1_noise005**	**Diff1_noise01**	**Diff1_noise015**	**Diff1_noise02**	**Diff2_noise005**	**Diff2_noise01**	**Diff2_noise015**	**Diff2_noise02**
SPC	94	96	95	96	96	92	92	90	95	95	83	58	97	96	85	43	96	97	62	15
V1-H	66	69	71	64	85	72	89	80	32	28	34	15	15	38	28	36	54	51	55	59
V1-S7	63	59	77	61	73	74	72	72	60	58	43	42	26	47	43	30	49	49	41	50
V2-H	96	95	95	89	75	62	65	39	63	78	64	50	50	39	32	33	49	64	52	48
V2-S7	94	93	90	87	86	47	44	28	94	77	64	27	49	42	33	24	49	62	56	47
V3-H	96	96	95	87	85	67	61	46	82	79	52	46	45	39	36	22	48	52	51	48
V3-S7	96	96	93	85	78	47	52	35	62	77	67	41	44	48	33	30	48	63	49	54
V4-H	96	96	97	95	95	93	90	89	62	67	31	27	30	37	43	48	56	52	60	63
V4-S7	95	81	94	95	59	78	51	57	57	60	31	51	34	50	52	25	50	58	53	21
V4-H abs	96	96	96	96	95	92	88	83	60	41	14	39	28	37	31	36	57	44	53	63

Symlet 7.

Since all the V1-4 versions of the on-line algorithm present the NEO as SD stage, it is worth to evaluate how much this stage influences both detection and sorting, compared to a solution based on a fixed threshold. To this aim, it has been limited the analysis to V4 with the *Haar* wavelet, compared to SPC (using an absolute threshold on a bandpass filtered signal).

The behaviour of the two solutions is very similar, the number of FP per minute being slightly better for the SD with NEO (Fig. 4.9). In terms of matching, as can be seen comparing SPC, $V4-H$ and $V4-Habs$ rows in Tab. 4.1, the results exploiting the NEO in the spike detection are better than those achievable with a fixed threshold.

From the results achieved on the synthetic database, it is possible to see how the adoption of 4 levels for the WD, with a *Haar* wavelet, removing both the approximation and the first detail, exploiting the NEO for the SD and performing the SS on the output of the WD leads to the best results. In a comparison with a top off-line SS algorithm such as the SPC, the results are similar until the correlation between the action potentials belonging to different neurons is lower than the level used in the on-line SS. However, the on-line WD and SD approach is superior to the one adopted in the SPC.

It is then worth to see whether such results reflect what happens on the real PNS signals, the target of the original algorithm [16]. Tests have been performed varying both the length (*len*) of the templates and the time frame for a pattern (t_L). Performance were evaluated comparing the algorithm implemented in [69] with the current V4 version. Figure 4.10 shows the accuracy obtained at the end of the SVM classifier testing phase on the real signals. Accuracy has been computed as average over 3000 different training/test partitions of the same dataset, as described in Sect.4.4. The trend for the V4 version is less steep, determining a minor influence of the t_L values, allowing to obtain a more efficient hardware implementation for the same classifier performance. Moreover, it is possible to note that the classification accuracy is always higher for the V4 solution, from a minimum of about 74% to a maximum of 96% with a standard deviation that decreases as t_L increases, regardless of the *len* value in the considered range.

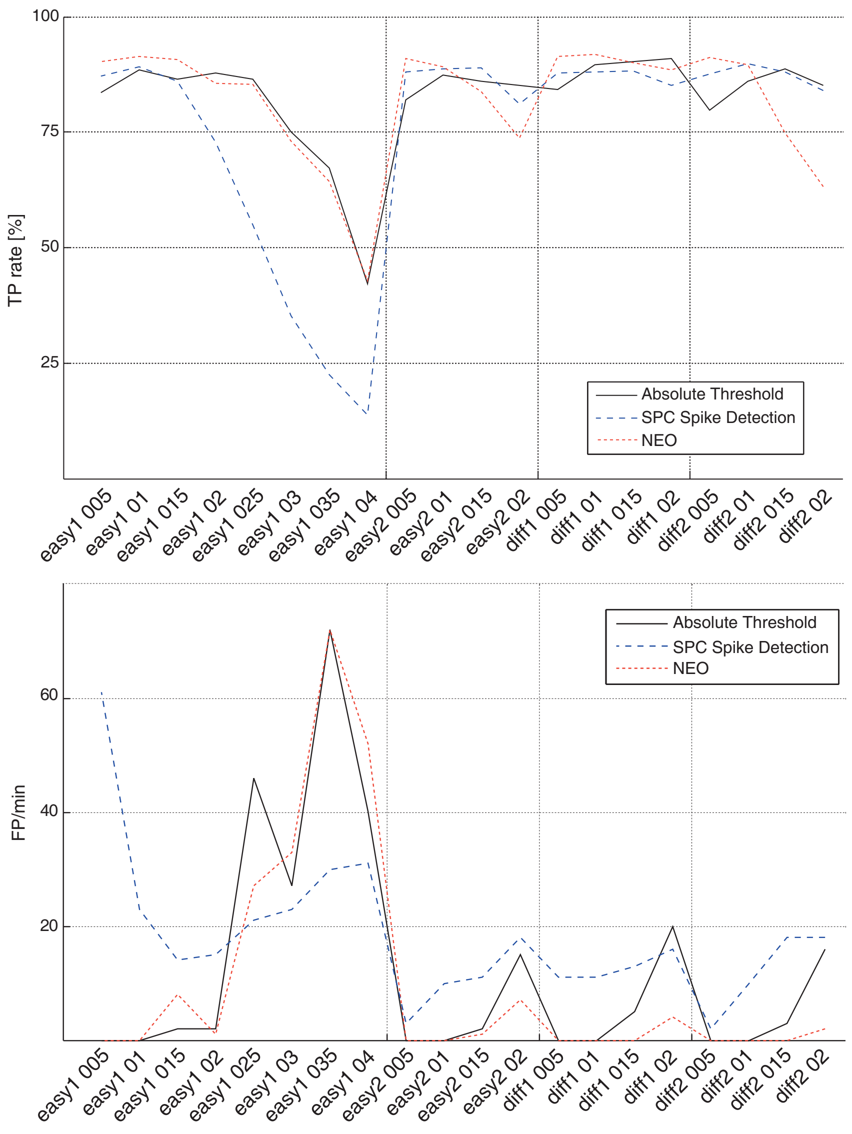


Figure 4.9: Performance comparison of the various spike detection approaches in terms of TP rate and number of FP per minute.

4.5.2 Efficiency analysis

The code of the best performing V4 version of the proposed algorithm using the *Haar* wavelet has been profiled in order to derive a latency model. It is a mathematical model, descending from cycle-accurate profiling, able to provide (with some approximations) the expected latency (CPU clock cycles) for a given code under different test conditions. Since the code optimizations, and consequently its performance, can be different if only a part of the code is compiled, in order to create a fair latency model the code was not modified (except for the parameters to tune) for the analysis of the different sections but the input data changed in order to be able to trigger specific behaviours. Compared to other techniques, this approach is very time consuming but allows pursuing accuracy.

The main issue in the creation of the latency model is the high number of branches that the algorithm can take during the different phases of the execution. Referring to Fig.4.4, it is clear that different latencies will be experienced when only the SD is performed or when

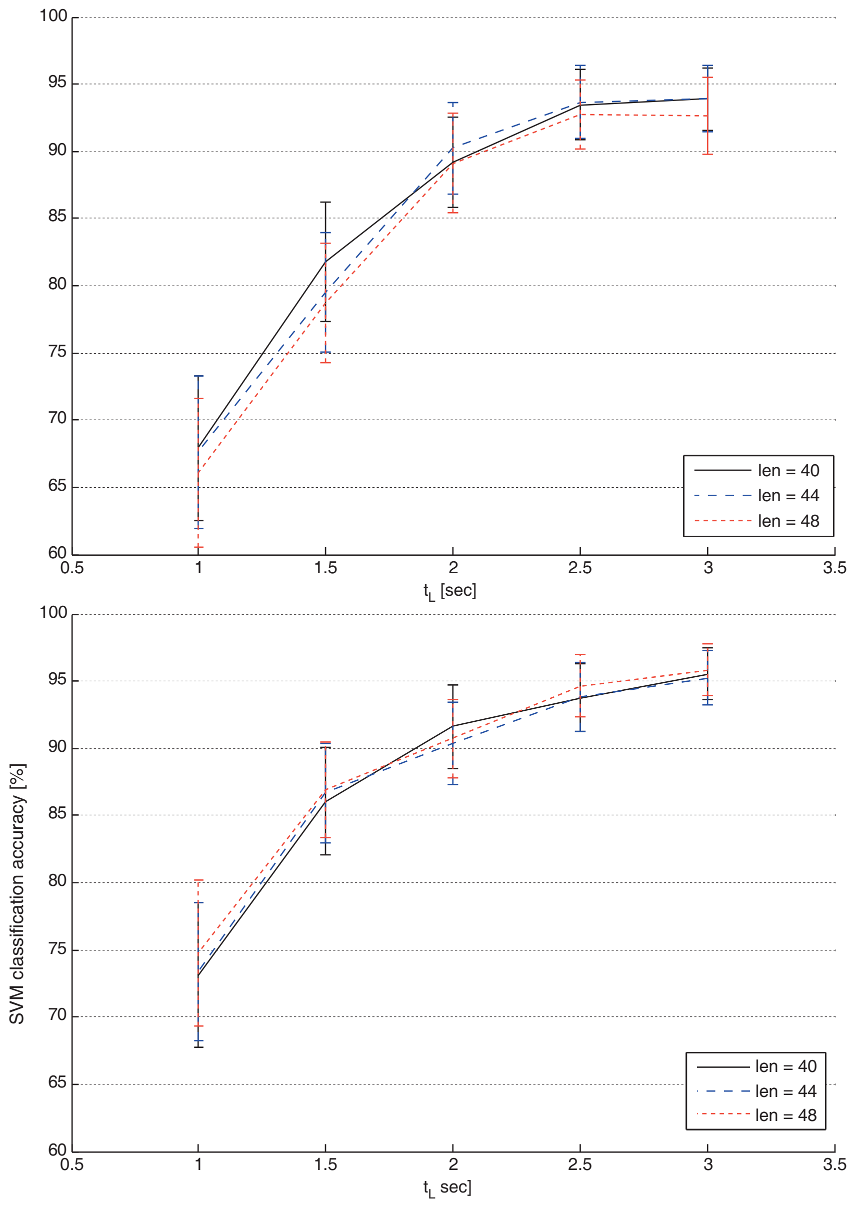


Figure 4.10: SVM classifier accuracy (mean and standard deviation over 3000 random training/test pairs on the same dataset) varying *len* and t_L parameters, comparing the algorithm implemented in [69] (top) with the current V4 version using the Haar wavelet(bottom).

the system is creating the templates (TPh) or during normal operation (SPh). Even if it has been created a model for each possible combination, varying the proper parameters, only the 2 most important working conditions will be analysed hereafter. The worst case in the TPh is identified in Fig.4.4 with the darkest shading, whereas the branches including the lighter shading are those related to the SPh. White blocks can be considered not part of any worst case scenario. Only the parts of the code that must be executed in real time have been evaluated for the latency model: the CL training and the templates reduction have been profiled apart.

The memory allocation strongly influences the code performance. All the code sections working on the data @12kHz, and the related variables, have been placed in L2 (configured as all-RAM) along with the on-line CL code @4Hz, whereas both the code and the variables used for the CL training have been placed in L3. The real-time code requires only 49kB of memory and the data in L2 expressly instantiated in the code require 114 kB of memory. Taking into account also the other code sections (used by the DSP/BIOS operating system of the DSP and its variables), the code on the internal RAM reaches 191 kB of the available 192 kB. The L3 memory usage reaches 1.28MB:259kB are reserved to the external data section for the CL whereas 26 kB for the related code section.

From the code profiling, the WD for the V4 version of the algorithm, with the chosen window length, requires about 380kcycles with the *Haar* wavelet. Such a result cannot be compared with that reported in [69], because the current memory requirements are higher and then it is necessary a larger exploitation of the external memory, with the consequent latency penalties. Within the current framework, the implementation of the original 3-level WD stage with the *Symlet 7* wavelet would require about 640 kcycles. Taking also into account the demonstrated superiority in terms of performance of the *Haar* wavelet compared to the *Symlet 7* one, the adoption of the former should be preferred. The final model during the TPh is described by the following relation:

$$c = \left[1.1 s_{len} + 1563 + \left(1562 M_{N_t} + 51.4\right) len + -31810 M_{N_t}\right] n_s + 0.6 len + 580280 \tag{4.9}$$

where c is the cycle count, s_{len} is the duration in samples of the spikes, M_{N_t} is the number of elements of the templates matrix, len is the template length in samples, n_s is the number of spikes in a b_len-sample window ($b\_len = 3000$ in this case). It is worth to note that the dependence from s_{len} is very small compared to the other parameters. For this reason, considering an average case of $s_{len} = 16.6$, as from the available signals, the final model during the TPh is:

$$c = \left[1581 + \left(1562 M_{N_t} + 51.4\right) len - 31810 M_{N_t}\right] n_s + 0.6 len + 580280 \tag{4.10}$$

Using this model it is possible either to:

- know the latency for a given n_s when the maximum number of templates in this phase is fixed to M_{N_t} and every template length is len or
- invert the model considering the maximum number of available cycles (Real-Time Bound RTB, which is 75Mcycles when clocking the DSP @300MHz) so that, fixing the structural parameters, it is possible to know how many spikes can be analysed in real time.

The second choice leads to the following model:

$$n_s = \frac{RTB - 580280 - 0.6 len}{1581 + \left(1562 M_{N_t} + 51.4\right) len - 31810 M_{N_t}} \tag{4.11}$$

Some of the implemented optimizations forbid the adoption of this model at a very fine granularity because, for instance, the parameter len must be a multiple of 4. The set of curves in Fig. 4.11 allows evaluating the maximum number of processable spikes per second, during TPh, as a function of M_{N_t}, for different values of len.

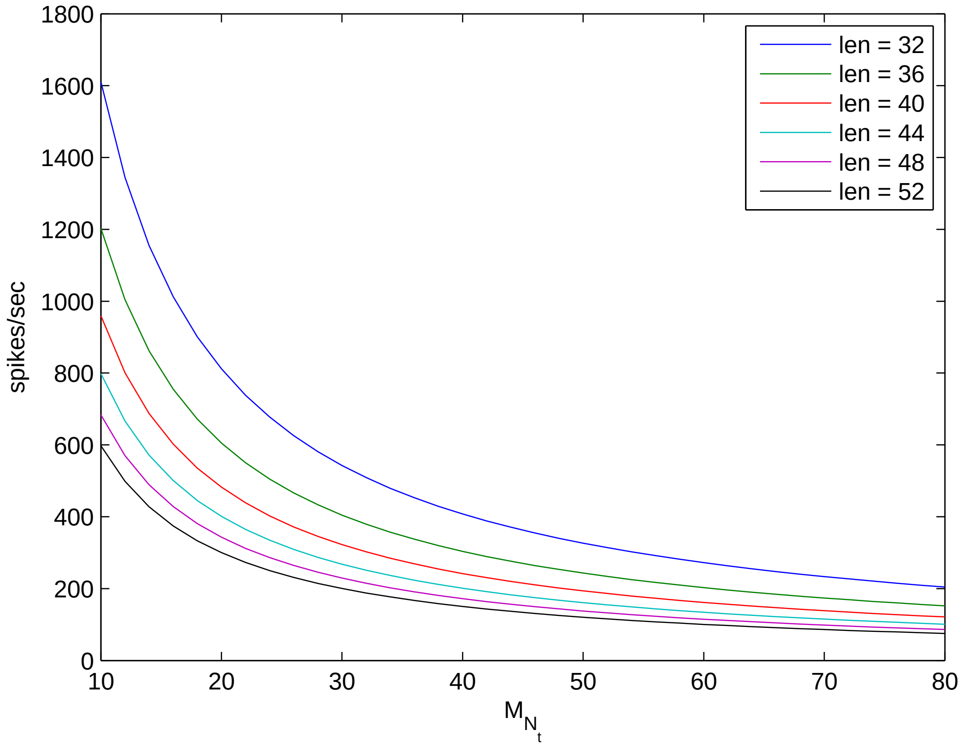


Figure 4.11: The maximum of processable spikes per second in TPh as a function of M_{N_t} (from top to bottom, the curves are drawn for increasing values of len).

Following a similar reasoning during the SPh, the final latency model is the following, where n_{feat} is the number of templates after their fusion (i.e. in the SPh) and n_{SV} is the number of support vectors.

$$n_s = \frac{RTB - 280n_{feat} - 1133n_{SV} - 574755}{(1562len - 31810)\, n_{feat} + 781} \tag{4.12}$$

The number of spikes per second in this case is depicted in Fig. 4.12, where n_{SV} has been fixed to a typical value of 700. With 40 templates in TPh and 10 in the SPh, and exploiting the same len of the SPC algorithm for the synthetic database (32 samples @12kHz), the on-line algorithm is able to process more than 400 spikes per second in TPh and more than 1600 in SPh, which represents an important result in terms of possible exploitation even in case of extension to multichannel recordings. In this case, the computational power can be distributed across the different channels with acceptable performance levels.

Summarizing, it has been developed the optimization of a state-of-the-art algorithm for PNS signals decoding in order to obtain the best performance on a limited-resources embedded platform such as an off-the-shelf DSP. Compared to custom VLSI or FPGA implementations, the adoption of these highly efficient micro-programmed architectures leads to a greater flexibility. For the time being, this is particularly useful for closed-loop experiments when the signal processing algorithms need to be quickly adapted to previous experimental evidences. The proposed analysis also identifies improvements to the original algorithm able to guarantee the real-time performance with improved quality in the results.

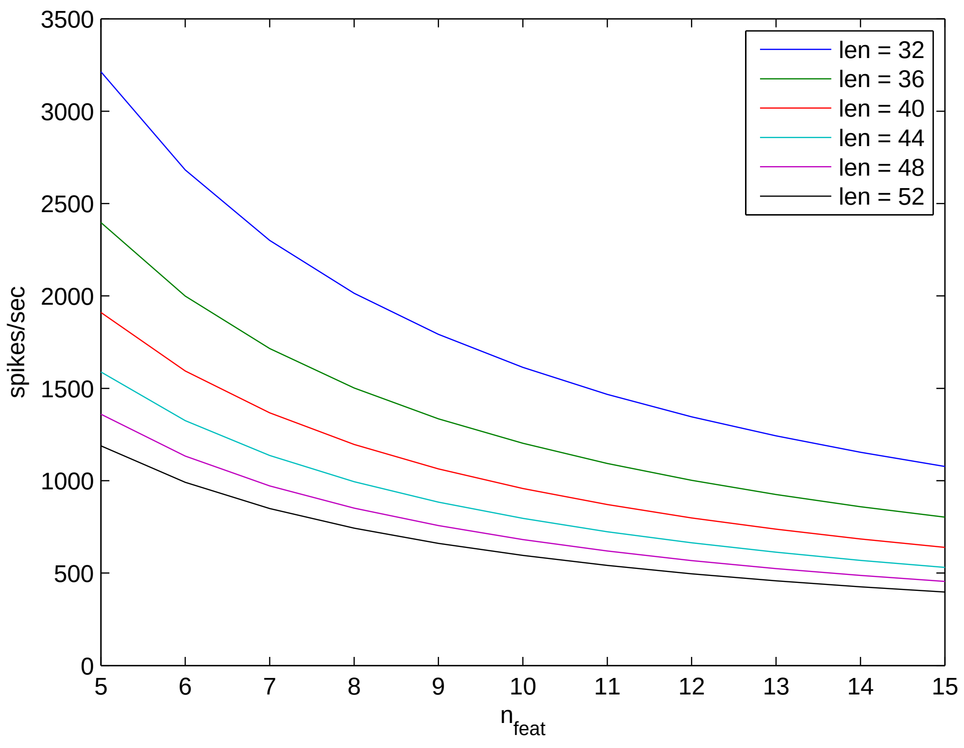


Figure 4.12: The maximum of processable spikes per second in SPh as a function of M_{N_t} (from top to bottom, the curves are drawn for increasing values of len).

Several tests have been performed both on synthetic neural datasets and on real afferent signals recorded in-vivo from rodents. The optimal version (V4) of the algorithm allows to achieve an accuracy up to 96% in classification, respecting what it has been achieved in [16] through an off-line processing based on the same algorithm, giving the possibility of performing the training of the prosthetic device autonomously by the patient himself without external tools. It uses 4 levels in the wavelet denoising stage and the simplest *Haar* mother wavelet rather than the commonly used *Symlet 7* one.

The main implementation issues regarding the porting of the whole processing algorithm on a floating point DSP platform, allowing the fulfilment of real-time constraints, have been discussed. A complex latency model has been derived, allowing the identification of the working point under different parameters setting. In a single-channel implementation, the algorithm is able to process up to 400 spikes per second when the unsupervised templates creation procedure is running, and up to 1600 in a use case where 10 templates are required to obtain the pattern features. Such numbers prompts the possible extension to a multi-channel scenario involving closed-loop real-time experiments including the complex phase of classifier training, now included in the same embedded framework so that also the training phase could be carried out without any external tool.

Chapter 5

VLSI Wavelet Denoising of Neural Signals

5.1 Introduction

The design of a portable/implantable low-power implementation for the real-time decoding phase of the peripheral neural signals for prosthetic solutions needs an efficient tuning process of the various processing stages in order to minimize the hardware resources necessary to correctly perform each of them, giving more attention to the wavelet denoising stage. With the increasing integration capabilities and the advancements in CMOS processes, it is progressively more common to require wearable or implantable electronics for such a processing. Applications like electrocardiography (ECG) [7] and electroencephalography (EEG) [11] often require these features. However, the low bandwidth of these signals and the relatively low number of channels (except for very specific applications like potential surface mapping or high density EEG) impose looser constraints compared to other applications like the neural signal processing under consideration.

As detailed in the previous chapter regarding to the first DSP implementation, wavelet denoising represents a common preprocessing step in case of low signal-to-noise ratio, as it happens in real operating conditions. It is especially fundamental for prosthetic applications to remove the background noise and the low-frequency EMG interferences related to the muscle stimulations placed near the electrodes added to the significant signal in order to obtain good performance at downstream of the next processing stages. When the real-time requirements are joined to the fulfilment of strict constraints about area occupancy and power consumption to allow the development of portable/implantable devices, only Application Specific Integrated Circuit (ASIC) implementations can be adopted.

Compared to micro-programmed solutions involving the use of microcontrollers or low-power digital signal processors (DSP), ASIC design requires highly specific skills and a longer development time [61], unfortunately leading to non-flexible architectures. This has been verified in [61] where pros and cons of FPGA and DSP solutions for a different wavelet-based denoising algorithm have been carefully evaluated. At the end of the analysis it has been achieved that the possibility of using a greater operating frequency on FPGA allows to use this type of implementations for biomedical applications which need high values of sampling frequency in a multi-channel scenario, compared to what it is possible for a DSP which has a fixed miprocessor architecture, without taking into account power constraints. On the contrary, it is fundamental to verify if a low-cost FPGA device using a typical fixed-point representation permits to obtain a satisfactory accuracy for the considered application avoiding performance degradation.

Some tools for automatic creation of hardware description language (HDL) designs have been presented to address such issues (e.g. ORCC, http://orcc.sourceforge.net/ [64] or the commercial Xilinx System Generator, http://www.xilinx.com/tools/sysgen.htm). Even though unable to significantly address the low-power requirements, these tools provide the possibility of exploiting an alternative language/method rather than HDL in the design phase [67]. The integration with well-known tools such as Simulink enables a faster development and, leveraging on parameterized HDL libraries, good performance in terms of area and power can be also achieved. For these reasons, in such a way every part of the algorithm must be carefully tuned quantifying the consequences of each of the possible design choices for an optimal hardware implementation in terms of necessary resources and performance degradation. In this part of the research activity, the overlooked phase of threshold estimation has been deeply analysed exploiting the Xilinx System Generator tool for the design of the architecture and the realization of hardware/software cosimulations, as in [5] where it has been used to remove power-line interference from ECG signals, with the aim of implementing efficient solutions from the hardware point of view allowing the realization of portable solutions.

This commercial tool has been chosen for its rapid prototyping advantage, evaluating the main performance figures associated to the hardware implementation of the wavelet denoising algorithm used for neural signal processing [18]. The ASIC implementation of this algorithm would be particularly useful in a multi-channel neural signal processing context, especially for neural prostheses [16] which must be able to work exploiting batteries as power source or in case that the WD stage should be placed nearby the acquisition system. At the same time, it allows highlighting the possible pitfalls hidden in the straightforward creation of an architecture from its typical algorithmic version. In particular, the threshold estimation stage is marginally considered in the largest part of the applications, usually exploiting fixed precomputed thresholds [41] or preferably opting for the estimation of the standard deviation of the noise by the Median Absolute Deviation (MAD), known to be a robust estimator of the dispersion in presence of outliers. Such an approach has been challenged in order to evaluate how it could lead to very inefficient or even infeasible architectures, not compliant with the requirements of the target application.

5.2 Algorithmic solutions for threshold estimation

Due that the implementation details related to the adopted wavelet denoising technique have been properly described in the previous chapter, in this part of the work it will be paid more attention on the methods for calculating the thresholds applied at decomposition downstream. The choice of the threshold influences the quality of the denoising so much that even data-specific approaches have been presented so far [54]. It can be fixed [41] based on empirical considerations or adaptive [73] according to the characteristics of the signal of interest, the same or different for all the detail signals at output of the various high-pass filters for each stage of the decomposition trellis. In particular, adaptive thresholds are typically computed estimating the root mean square rms or the standard deviation σ of the signal at the different levels and then correcting it by a scaling factor. Different multiplicative factors have been derived and are preferred by different authors, as for the Minimax [16], Stein's Unbiased Risk [51] or Universal [6] methods. Due to the robustness to the presence of outliers, usually the preferred method is to estimate σ by the median absolute deviation (MAD), defined as:

$$MAD = median_i \left(|X_i - median_j (X_j)| \right) \tag{5.1}$$

Thanks to the high-pass nature of the detail signals, it is common to implement the MAD as simply the median of the absolute value of the details:

$$\overline{MAD} = median_j \left(|X_j| \right) \tag{5.2}$$

It is possible to prove that $MAD \approx 0.6745\sigma$. Nevertheless, the MAD is preferred for the aforementioned robustness, especially in neural signal processing where the neural spikes can be considered as partly composed of outlier samples in recordings with a good signal to noise ratio. Such an approach, which is perfect for off-line processing, overlooks the real computational complexity of the median operator in case of continuous adaptation. This approach has been challenged by some authors trying to develop efficient implementations for different biomedical signal processing applications [69, 93]. In the following, the adoption of the MAD and the sample standard deviation on a sliding window has been compared not only in terms of denoising quality, but also in terms of feasibility in the perspective of a low-power real-time implementation as needed for an implantable unit for the control of a neuroprosthetic device [16], using as a testbed a prototypical FPGA implementation programmed exploiting the Xilinx System Generator tool.

5.3 Architecture Design exploiting Xilinx System Generator

As starting point, it has been considered even in this case the translation-invariant wavelet denoising trellis according to what defined by the *Ã -trous* algorithm [80]. In order to keep low the resource requirements in the perspective of an FPGA implementation, the cost-effective *Haar* mother wavelet has been chosen even remembering that it requires very small filters and allows to achieve good performance combined with the next processing steps of the decoding algorithm for prosthetic applications as verified by the previous DSP solution. Having an input signal sampled at $12kHz$, with 4 levels and removing the last approximation signal, an overall high-pass behaviour able to reject the low-frequency interference components below $375Hz$, outside the bandwidth of neural signal, can be obtained. In order to evaluate the impact of the thresholding stage, no specific optimizations have been made at the level of the filter banks for decomposition and recomposition. The low-pass and the high-pass FIR filters have been implemented in the Transposed Direct-Form I, inserting registers with delay equal to increasing powers of 2 between the internal adders to perform the required oversampling process in the different stages.

As it has been already said before, the choice of the threshold can have an impact on the quality of the denoising. However, the repercussions in terms of hardware requirements need to be carefully evaluated in the perspective of a low-power implementation targeted on the reference prosthetic application. In this application, two alternative solutions have been considered which are based respectively on the calculation of:

- the MAD of the signal, using either a combinatorial or an iterative approach;
- the sample standard deviation σ of the signal.

As scaling factor, the Universal one has been adopted, so that:

$$\theta = \frac{\overline{MAD}}{0.6745}\sqrt{2\log M} \tag{5.3}$$

in the first case, and:

$$\theta = \sigma\sqrt{2\log M} \tag{5.4}$$

in the second one. M is the length of the signal frame in terms of number of samples.

In order to provide adaptiveness in an on-line scenario based on the characteristics of the processed signals, both the first and the second solution have been adapted to work on a sliding window of variable size, with an overlap of three quarters of the overall window length. In the first case, this requires sorting the new window every time in order to extract the median, however in the second case a faster solution can be implemented taking into account only the remaining quarter of the last samples window. Starting from the technique used in [69] and thanks to the zero-mean nature of the high-pass detail signals, for each N new input samples in the window, the related sum of squares for the *j-th* decomposition level is computed as:

$$s_j = \sum_{n=1}^{N} d_j^2[n] \tag{5.5}$$

and then used to determine σ for the 4 times larger windows as:

$$\sigma = \sqrt{\frac{1}{4N-1}\sum_{k=1}^{4} s_j} \tag{5.6}$$

According to the sliding window approach, the threshold value is updated every N sampling periods taking into account the 4 times larger observation window ($M = 4 \times N$). The longer the observation window, the better the estimation accuracy, provided that instantaneous variations (neural spikes) do not influence the threshold computation which must represent the current noise level added to the significant signal. Such a processing can be delegated to a host processor picking up the detail samples at the high-pass filters output and computing the various thresholds. However, when a non-microcoded solution is pursued, because of the need to fulfil the real-time and low-power constraints, threshold estimation can be performed by dedicated cost-effective hardware. In this case, the complexity depends on both the chosen threshold estimation method and the length of the observation window M.

In order to evaluate the different solutions from the hardware perspective, starting from a high-level model with an acceptable complexity even for researchers not accustomed to HDL modelling, it is possible to use tools such as Xilinx System Generator. In this way, the user-friendly environment provided by Matlab Simulink can be exploited both to create the hardware design and to perform accurate co-simulations taking into account the hardware implementation of a part of the system under test, as in the example shown in Fig.5.1.

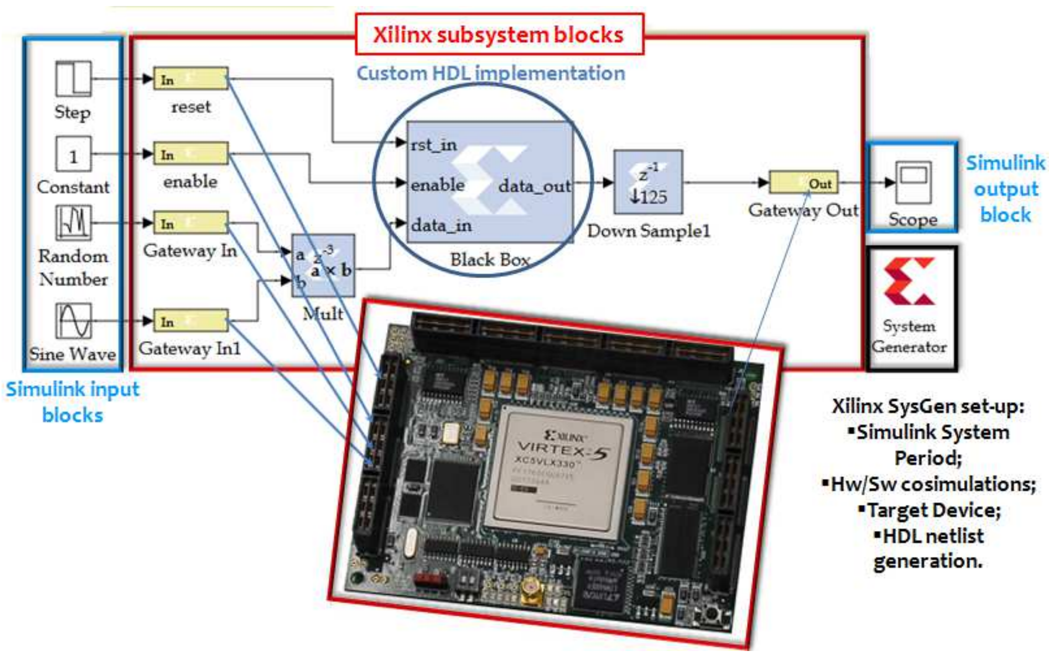


Figure 5.1: Example of a system developed by Xilinx System Generator.

The tool allows to choose whether to use the hardware blocks coded by the user (providing the HDL file) or those provided by Xilinx. The latter only require to define the internal signal representation (fixed-point, unsigned or signed as 2's complement, etc.) as for the desired functionality, whereas the former must adhere to a standard interface mainly requiring an enable signal for each input clock to synchronize the modules inside the model. When the design has been completely created, the set of the hardware blocks can be mapped on a real FPGA board in order to evaluate the percentage of used resources and the behaviour by hardware/software co-simulations. This is very useful to accelerate the simulation time compared to the totally software case.

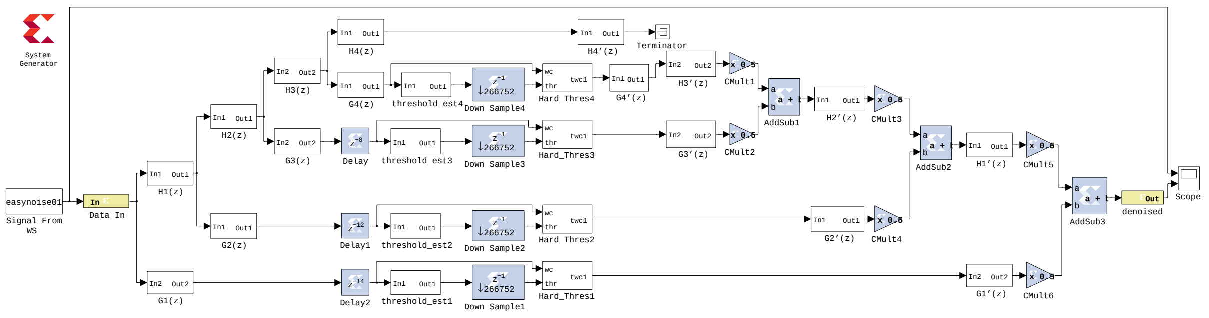


Figure 5.2: Simulink model of the wavelet denoising scheme using System Generator blocks.

The wavelet denoising algorithm, created exploiting the Xilinx System Generator tool, is shown in Fig. 5.2. The *Gateway In* and the *Gateway Out* blocks delimit the hardware part of the Simulink model, defining the interface signals to be mapped on the various pins available on the FPGA. The System Generator block fixes the co-simulation parameters, the Simulink

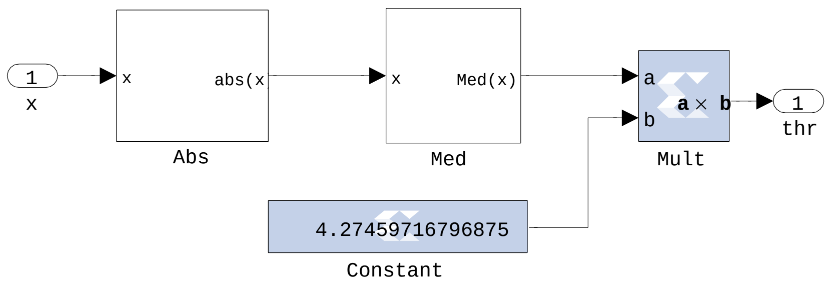


Figure 5.3: Simulink Model of the Threshold Estimator block with the constant defined as for the case of M=64 samples per window.

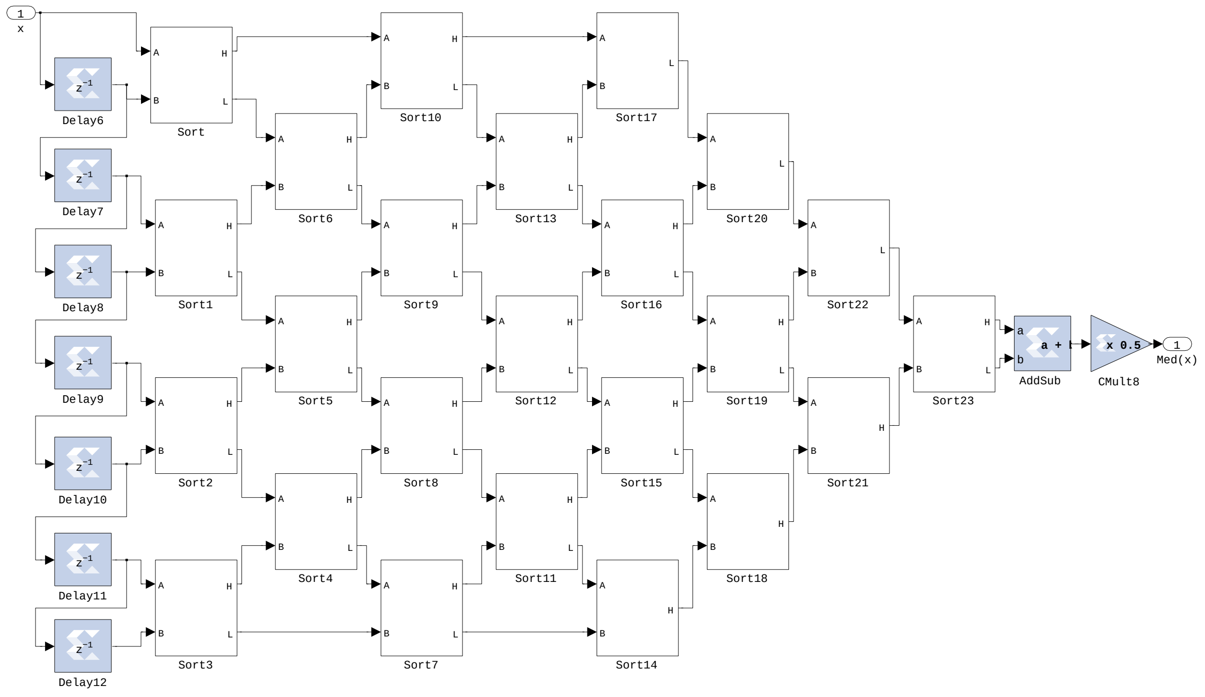


Figure 5.4: Simulink Model of the sorter using an unfolded combinatorial approach for windows with M=8 samples.

system period, the target board to map the hardware sub-model, and so on. In these tests, a Xilinx FPGA Virtex-5 LX330 has been chosen for its considerable amount of available resources.

In case the $\overline{MAD}$ is used, the System Generator implementation involves the extraction of the absolute value of each input samples, the computation of the median value of the incoming windows of M input samples and the multiplication by a constant, as defined in (5.3). The corresponding Simulink model is depicted in Fig. 5.3. The median value calculation requires the hardware implementation of a sorting algorithm which represents a costly operation from the hardware point of view.

A first possible solution can be the unfolded sorter presented in [6], for which the Simulink model considering windows of $M = 8$ input samples is presented in Fig. 5.4. The basic sorting cell makes the comparison between two inputs A and B and swaps them if $A < B$. It is possible to demonstrate that, if the comparators work in parallel, $M-1$ steps are sufficient to properly perform the sorting of M elements. The output is updated in a combinatorial way every time a sample arrives in input at the sampling frequency f_s, after the proper shift of the values saved into the registers needed to prepare the input samples for the processing. The $\overline{MAD}$ is computed as the arithmetic mean of the two central elements of the sorted array for an even number of samples.

This solution presents several pitfalls from a hardware implementation perspective. In particular there is a clear scalability issue related to the enlargement of the observation window. In this case, the increasing internal critical path determined by the cascade of comparators limits the maximum operating frequency, beyond the penalty associated to the huge amount of hardware resources.

To overcome such problems, an iterative (folded) approach to the sorter able to reuse the same resources at each step, similar to that proposed in [53] about the Burrows-Wheeler transform but adapted to the wavelet denoising case, can be used. In this case, the sorting strategy uses only two levels of comparators. At the beginning, the swaps are performed only for the registers related to odd adjacencies, activating only the first level of comparators. If the vector is not yet sorted, at the next iteration only the comparators of the second level are active, and so on until the sorting process is completed. It is possible to demonstrate that the number of necessary steps is $M/2$ if M is the number of samples to sort. Figure 5.5 shows the iterative scheme.

The *swp* signal coming out from the comparator block is used to specify that the two inputs have been swapped. The samples in input to the parallel sorter, belonging to each observation window, are temporarily saved into a single-port memory. Immediately after the last sample of the window has been saved in this memory, its content is copied into the registers and the sorting process can start. When all the *swp* signals are equal to 0 during the last iteration, the input vector is correctly sorted. A finite state machine, one for each *Threshold Estimator* block (i.e. one for each decomposition level), is used to control the various phases of the process.

In order to compare the hardware characteristics of these models of threshold estimation based on the $\overline{MAD}$ against those of a traditional sample standard deviation as described above, an hardware model has been designed also for such an approach. Fig. 5.6 shows the Simulink model for this implementation. Every time an input sample arrives, it is squared and added to the current value of s_j. After N samples, the final value of s_j is saved in one of the 4 locations of the single-port RAM used as circular buffer in order to determine the correct value of σ over the sliding window.

Regardless the chosen approach, the value of the threshold θ is sent in input to the *Thresholder* block, able to apply the hard thresholding on the detail samples. It should be also considered that the value of θ is different for the various levels.

5.4 Experimental Results

Before analysing the results in terms of hardware resources which are necessary for the different solutions presented above, such solutions have been evaluated from a functional perspective. To this aim, the same publicly available dataset of simulated neural signals [72] described in the previous chapter has been exploited as input test data. The synthetic signals are obtained by linearly mixing an artificial sequence of real spikes from three neurons to other spikes at random times and amplitudes, representative of the background activity

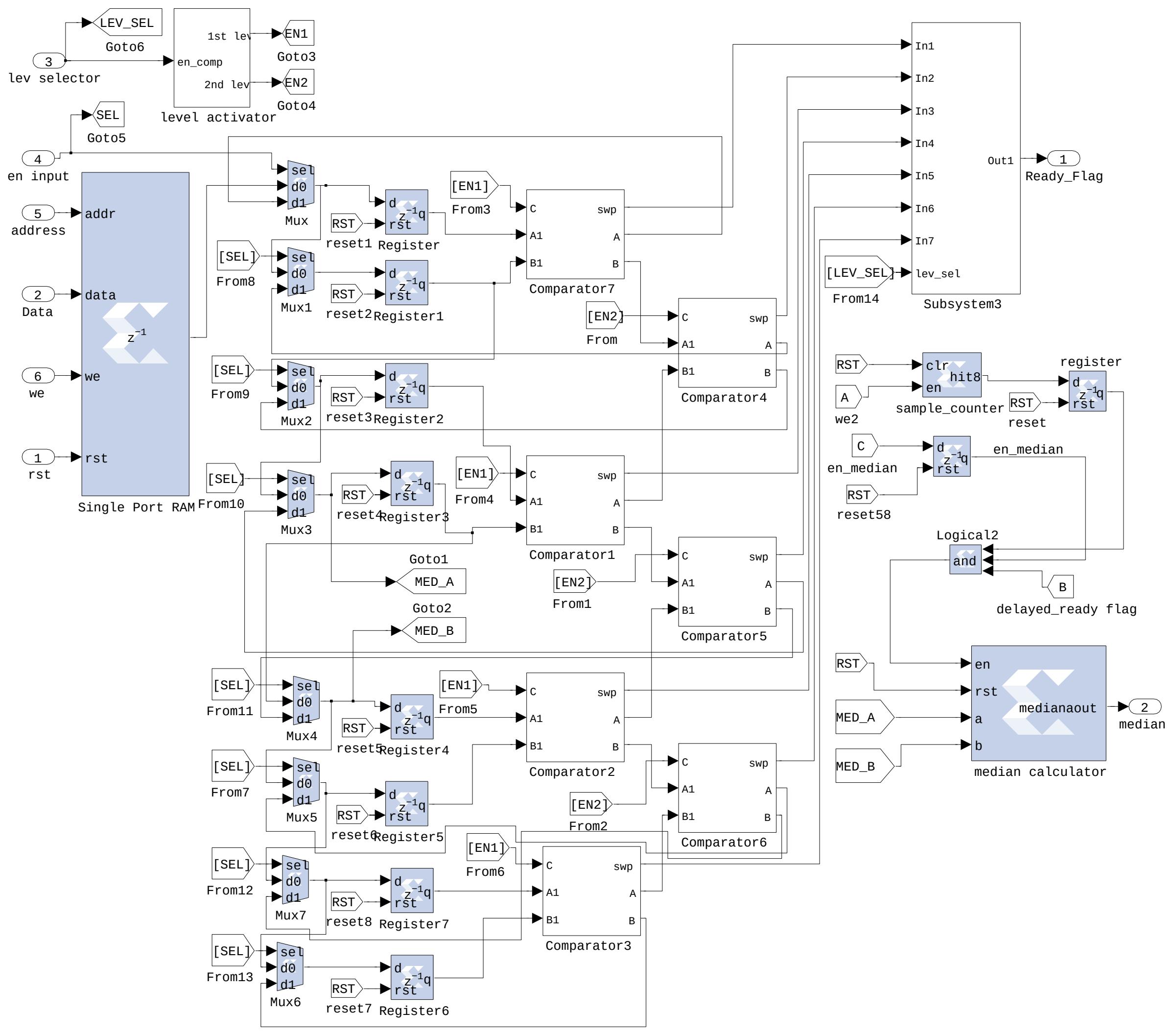


Figure 5.5: Simulink Model of the sorter using an iterative approach for windows with M=8 samples.

(of tunable intensity) of the neurons at a greater distance from the recording electrodes. The sampling frequency has been scaled to 12 kHz and the useful bandwidth is declared to be in the range [300Hz - 3kHz].

5.4.1 Functional Evaluation

The different versions of the whole wavelet denoising system have been mapped on the target FPGA in order to evaluate, by hardware-software co-simulations, the system performance under real conditions. Fig. 5.7 shows in the first row the neural signal with a low level of background noise used as input for the two hardware implementations based on the calculation of the $\overline{MAD}$ and of the σ (the two versions of the one implementing the $\overline{MAD}$ produce the same results). The next rows present the related outputs considering observation windows of $M = 4 \times 64$ samples.

It is possible to see that for both solutions, the wavelet denoising is able to remove, after an initial transient, the background noise added to the neural signal without cutting signif-

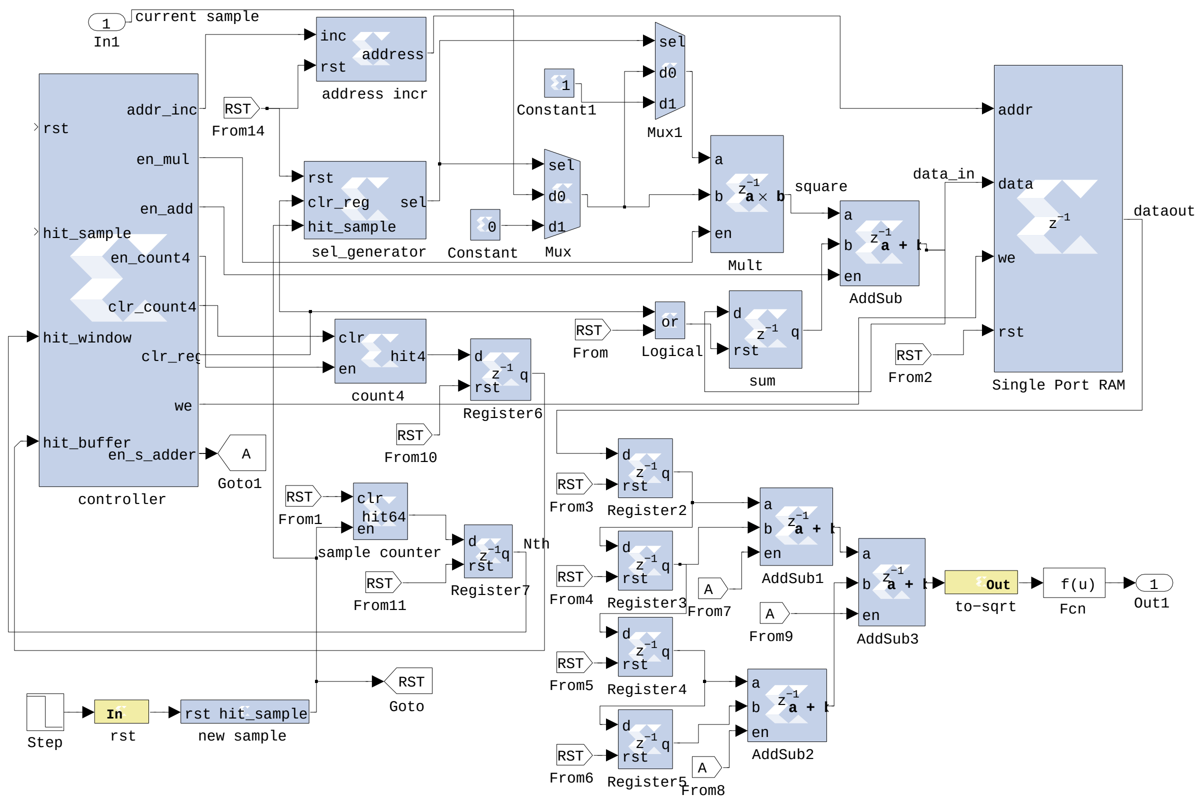


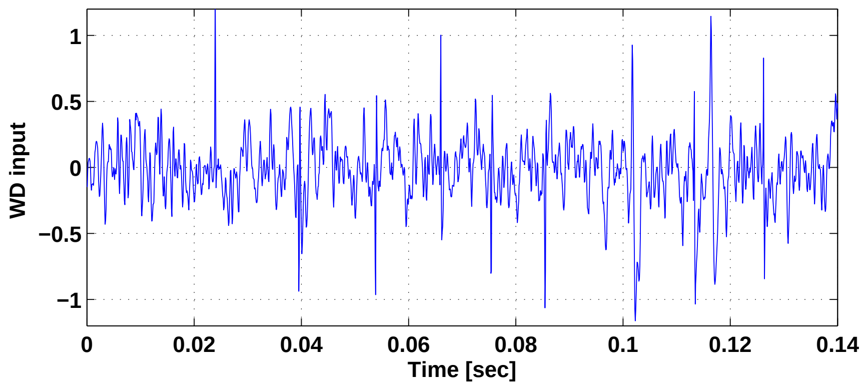
Figure 5.6: Simulink Model of the Threshold Estimator block based on the calculation of the sample standard deviation.

icant spikes. The same performance can be achieved using the same input signal but with a stronger background noise for which it is difficult to identify the various spikes on the raw signal, as can be shown in the first row of the Fig. 5.8. Even using neural signals with very low SNR, the two implementations behave similarly preserving the relevant spikes.

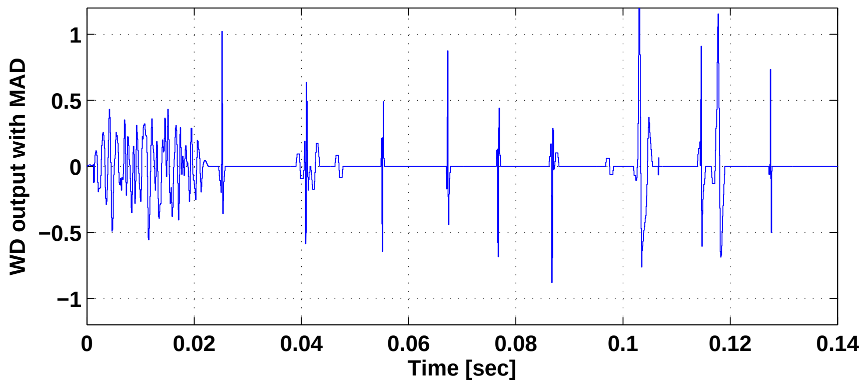
Then, the possibility of enlarging the observation window has been considered in order to provide a more significant frame for computing the statistics on the signal. For example, Fig. 5.9 shows the outputs of the two solutions using $\overline{MAD}$ and σ in the case of windows of $N = 128$ samples and a low level of background noise. The initial transient is obviously longer in comparison to the previous cases.

It has been also analysed the trend of the thresholds in output from the same decomposition level for different observation window lengths, for the two hardware solutions. The aim is to verify which is the minimum value of N, considering a sliding window length of $4 \times N$, that allows obtaining good performance in denoising.

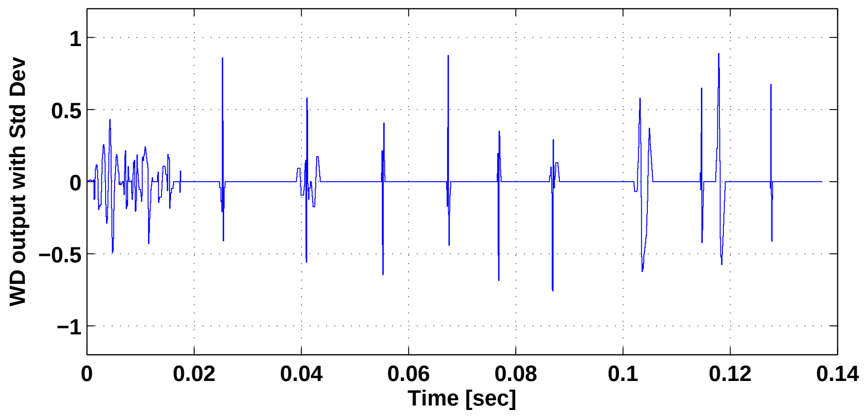
As can be noticed from Fig. 5.10, after a variable transient period according to the chosen value of N, the longer the observation window the better the stability of the threshold, not influenced by the presence of the neural spikes of interest. In fact, in the case of $N = 128$, the threshold estimation assumes an almost constant trend; the goal should be that of selecting the solution which provides the best compromise in terms of threshold estimation and required hardware resources.



(a) Input Signal



(b) WD output using the $\overline{MAD}$-based threshold

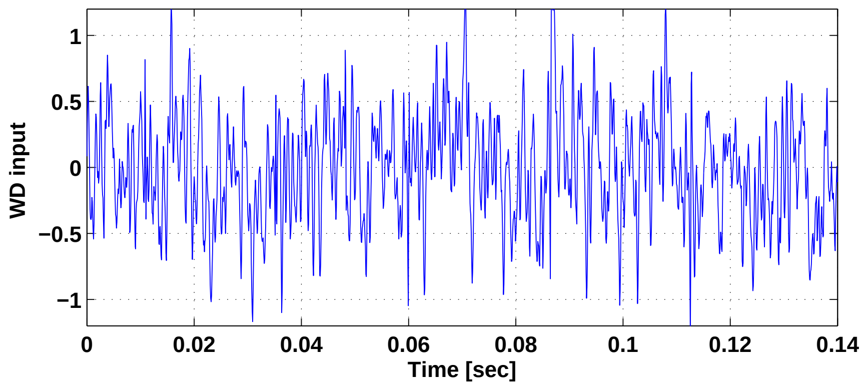


(c) WD output using the σ-based threshold

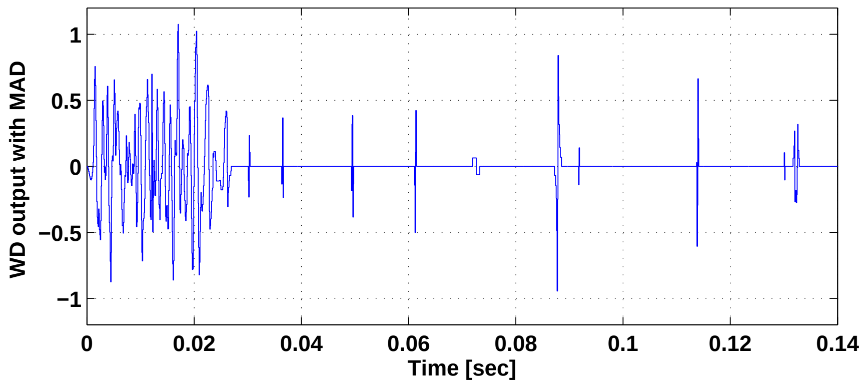
Figure 5.7: Wavelet Denoising input and outputs: low level noise, N=64 samples.

5.4.2 Hardware Figures of Merit

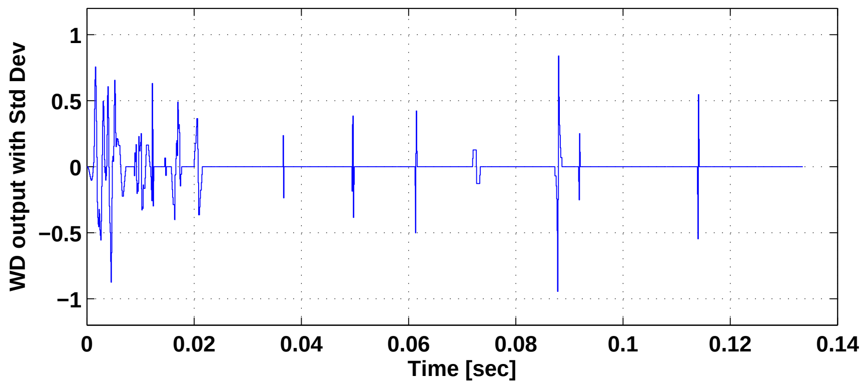
Thanks to the possibilities offered by Xilinx System Generator, also to map the implemented designs on real hardware, in this case the Xilinx FPGA Virtex-5 LX330 device, pros and cons in terms of necessary hardware resources have been evaluated for the different solutions presented above. The final goal is to determine which is the best solution allowing to achieve a



(a) Input Signal



(b) WD output using the $\overline{MAD}$-based threshold

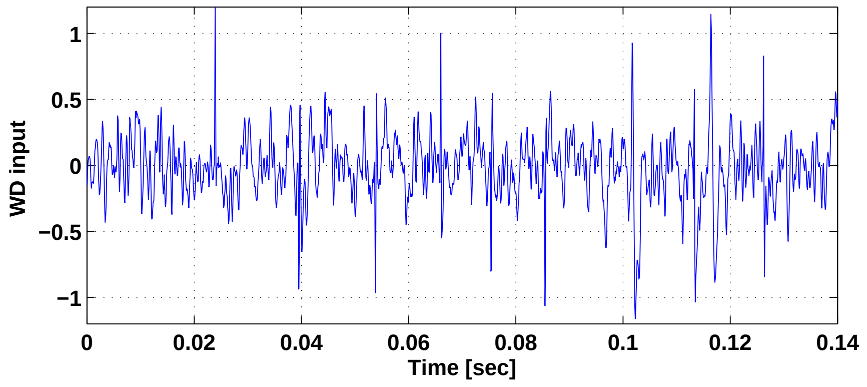


(c) WD output using the σ-based threshold

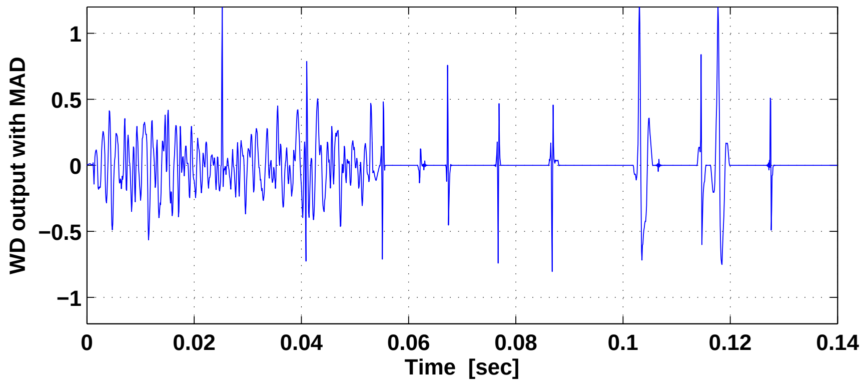
Figure 5.8: Wavelet Denoising input and outputs: high level noise, N=64 samples.

good accuracy with limited area and power consumption, in the light of the realization of a VLSI chip implementing such a processing stage for an implantable unit in the neuroprosthetic field.

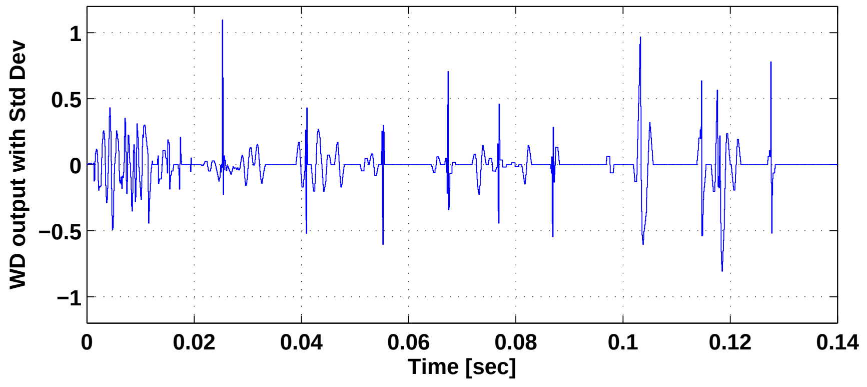
Synthesis results are presented in Table 5.1, which shows the percentage of available slices and Look-up Tables (LUTs) needed for the three considered threshold estimation blocks only, since the remainder of the wavelet denoising implementation is the same regardless of this stage.



(a) Input Signal



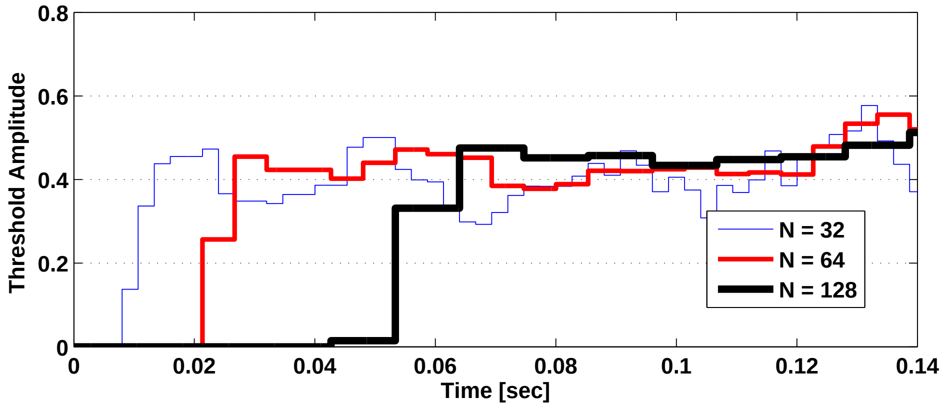
(b) WD output using the $\overline{MAD}$-based threshold



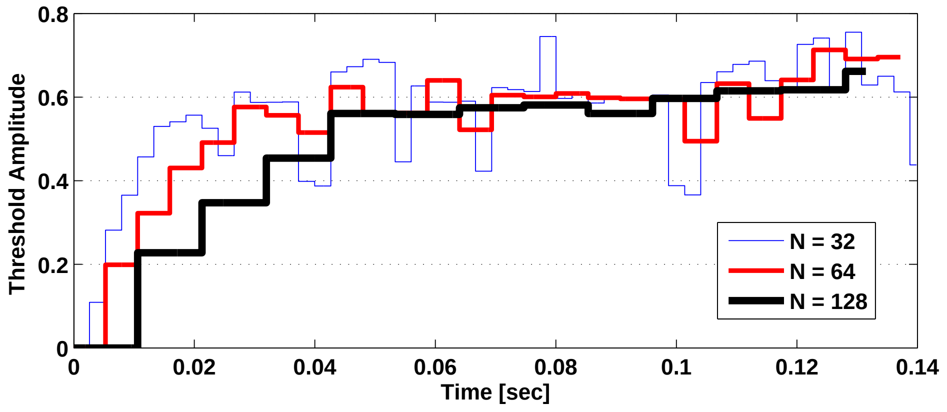
(c) WD output using the σ-based threshold

Figure 5.9: Wavelet Denoising input and outputs: low level noise, N=128 samples.

The solution based on the combinatorial (unfolded) $\overline{MAD}$ implementation, as highlighted in Table 5.1, is absolutely inefficient, taking into account that it has been presented only for $N = 8$ with the usual 4-times larger observation window. A rough estimation of the hardware resources required in case of $N = 32$ would lead to more than 330kLUT over the 207360 available ones, thus exceeding the considerable amount of physical resources on the target FPGA device. The huge amount of LUTs, compared to the folded version, is incompatible with a real implementation in the context of this application, taking into account that



(a) $\overline{MAD}$-based solution



(b) σ-based solution

Figure 5.10: Threshold variation over time using different lengths of the observation window

Table 5.1: FPGA synthesis results for the *Threshold Estimator* varying the length of the observation window.

	N	$f_{max}[MHz]$	Slice Registers	LUTs
σ	32	417.34	156 / 207360 (0.07%)	80 / 207360 (0.04%)
	64	416.61	157 / 207360 (0.07%)	82 / 207360 (0.04%)
	128	416.02	160 / 207360 (0.07%)	84 / 207360 (0.04%)
unfolded $\overline{MAD}$	8	417.08	558 / 207360 (0.27%)	20270 / 207360 (9.77%)
folded $\overline{MAD}$	32	242.78	2493 / 207360 (1.20%)	7164 / 207360 (3.45%)
	64	246.70	4932 / 207360 (2.38%)	14377 / 207360 (6.93%)
	128	221.42	9806 / 207360 (4.73%)	28861 / 207360 (13.91%)

the observation window length should be large enough to properly estimate the statistics of a signal sampled at 12kHz.

FPGA synthesis results demonstrate that the wavelet denoising solution based on the threshold estimation by the sample standard deviation allows minimizing the necessary hardware resources regardless the length of the observation window. Furthermore, the usage of slices and LUTs of the $\overline{MAD}$-based solution, even using a folded approach, is clearly incompatible with an efficient implementation of this processing stage, especially compared to the same data related to the σ-based implementations.

Summarizing, the choice of the threshold estimation technique, more than having an influence on the quality of the wavelet denoising algorithm, has significant reverberations on the feasibility and efficiency of a custom VLSI architecture aimed at an implantable chip in the context of neural prostheses. The comparison between a sample standard deviation and the widespread $\overline{MAD}$ has revealed similar functional performance with dramatically better characteristic of the former in terms of hardware implementation, regardless the $\overline{MAD}$ is implemented as a combinatorial trellis as suggested by some authors or in a more efficient folded version. This optimal approach can be thus exploited in order to develop the hardware implementation of the wavelet denoising stage with the minimum resources which can be mapped on an embedded system allowing portability and satisfying the timing and power constraints imposed by the target application.

Chapter 6

A Coarse-Grained Reconfigurable Approach for Low-Power Neural Signal Decoding

6.1 Introduction

Starting from the analysis proposed in the previous chapters and considering the aim of providing an implantable/portable implementation of the considered neural signal decoding algorithm for prosthetic applications, efficient resource management and specialized low-power design techniques must be adopted in order to fulfil tight real-time constraints with the use of the minimum hardware resources.

Extracting information in real-time from physiological signals to determine the intention of movements of an amputee and generating the relative electro-mechanical stimulations in input to a cybernetic hand represents a computational intensive task, especially in case of a huge number of channels. Fortunately, it is not the case of PNS interfaces compared to the CNS one for which the number of active channels is often quite limited. Nevertheless, given similar timing constraints imposed by the application, it is fundamental to exploit massive parallelism in order to minimize the execution latency, even that it comes at the expenses of a power profile not compatible with implantable battery-powered devices which must be placed nearby the acquisition system.

Several works have been presented so far in this field, the largest part of them proposing Field Programmable Gate Arrays (FPGAs) as implementation target to guarantee more parallelism than processor-based solutions [8]. However, FPGA potential flexibility (needed to adapt the algorithms to the ongoing experimental evidences) is in contrast with the complex traditional ways to design digital architectures. In fact, specific abstraction tools, such as those based on Simulink [24], cannot be used when the control on the low-level implementation details is pursued. Furthermore, such tools are not aimed at the maximum reuse of the basic building blocks (at a coarse granularity) so that their outcomes are not efficient in terms of area and power, assigning more responsibility to the designer in the creation of the HDL implementation.

For these reasons, it is proposed to adopt a novel approach in the specification and in the hardware implementation of the considered decoding algorithm, taking into account the various processing steps described in the previous chapters except the final SVM classification part. This approach is based on the use of an automatic tool, called Multi-Dataflow Composer (MDC), which allows to create a coarse-grained reconfigurable architecture with the minimum hardware resources exploiting the same Functional Units (FUs) to perform different parts of the algorithm. Moreover, the MDC approach is extremely attractive because the generated hardware platforms are flexible, fitting to parametric solutions adjustable at runtime based on the characteristics of the processed signal.

In [67] it has been demonstrated the applicability of this tool in the field of image/video processing. The MDC tool has been in fact conceived with studies related to the MPEG Reconfigurable Video Coding (RVC) [1] so that its application was straightforward. Nevertheless, the possibility of automatically managing the composition of reconfigurable platforms, power and area aware, makes it potentially suitable for other applications domains too, such as the biomedical one. It leads to a final implementation characterized by area and power saving that is mandatory for PNS implantable devices. The goals of this part of the work are the following:

- to exploit strategies of hardware reusability through runtime reconfiguration to perform consecutive processing steps in order to provide area and power minimization;
- to verify the orthogonality at application level of the MDC approach with respect to the original reconfigurable video coding domain;
- to test the proposed approach for the implementation of the real-time neural signal decoding algorithm creating an embedded system including a host processor and a reconfigurable coprocessor mapped on an FPGA;
- to evaluate the performance in terms of accuracy and typical metrics of a digital hardware design verifying the proper functionality with the same dataset of synthetic neural signals described in the previous chapters.

6.2 Exploiting the Multi-Dataflow Composer Tool

6.2.1 Multi-kernel datapath generation

Systems based on a reconfigurable approach are often called adaptive, in the sense that their logic functionality and interconnect can be customized to suit a specific application, by programming them at the hardware level. Dealing with reconfiguration, there are mainly two major levels of configurability:

- fine-grained, complete or runtime partial reconfiguration of the substrate (e.g. FPGA platforms);
- coarse-grained, reconfiguration of the interconnections among the involved functional units (FUs).

Fine-grained architectures are more flexible than coarse-grained ones but the counterbalance is that the overhead of passing from a configuration to another is huge, limiting the set of applications in which they can be exploited. Moreover, opting for a fine-grained approach, besides the shut-down state of operation necessary to change the context, it is necessary also to be able to afford a dedicated storage space to memorize the bitstream file needed to configure the various cells available on a FPGA. On the contrary, coarse-grained reconfiguration is still affected by the target physical space limitation, but it is able to provide faster switching at runtime between different hardware implementations: neither requiring any context change nor any need of specifying and storing fine-grained FPGA bitstreams. The latter approach can be useful in the case that an algorithm is partitioned into a sequence of computational kernels and its execution can be improved building up a reconfigurable platform able to switch among them during its execution.

For these reasons, the proposed MDC approach allows us to create a coarse-grained reconfigurable coprocessor able to map on a minimal set of FUs the functionalities required by the computational kernels identified in the target neural decoding algorithm. They must present commonalities at actor level to maximise the benefits in terms of area and power saving, where actors represent the high-level computing elements of a dataflow description integrating a particular functional unit. This type of description is widely used in signal processing, allowing to describe the required functionalities through the interaction of actors representing the nodes of the dataflow graph. It is particularly suitable when it is necessary to map in hardware only some parts of the whole application, generating a sort of computing accelerator at the disposal of other general-purpose processors.

These kernels are selected in the application flow diagram as the best candidates for a hardware implementation, characterized for being: repetitive, computationally intensive and with a reduced amount of conditional executions. Runtime reconfiguration and FUs reuse are allowed by the use of low-overhead switching modules (Sboxes) instantiated in the final platform activating different paths according to a specified kernel code which identifies the required computation.

Mapping kernels on a reconfigurable hardware substrate is not so straightforward as their number grows. Therefore, support tools are needed to speed-up this process. MDC [68] is a tool that can effectively serve the aforementioned purposes providing the automatic generation of reconfigurable coarse-grained platforms starting from the high-level dataflow descriptions of the identified kernels. Such description are given in the Network Language (NL), a dialect derived from XML and based on the Open Dataflow format (http://opendf.sf.net). An overview of the tool is shown in Fig.6.1.

The tool front-end (*Mutli-Dataflow CAL Composer, MDCC*) acquires the input specifications and combines them, identifying the common actors and properly inserting the Sboxes, into a multi-dataflow specification called *Directed Flow Graph* (*DFG*), able to implement all the given kernels of the application under consideration, preserving their computational correctness. The back-end (*Platform Composer*) maps the DFG into a coarse-grained reconfigurable hardware platform (*Global Kernel*) described in a Hardware Description Language (HDL).

To provide that such Sboxes can switch at runtime among the different datapaths without a low-level bitstream reconfiguration, they are controlled by dedicated Look-Up Tables (LUTs) whose content is automatically defined at design-time by the MDC tool. The LUTs are addressed by the kernel code and return the selection bits for the Sboxes. Such bits are used inside them to drive the internal multiplexer/demultiplexer in order to physically reshape the datapath accommodating the desired processing. Due to the low complexity of the switching modules, it is possible to change the implemented kernel in the generated

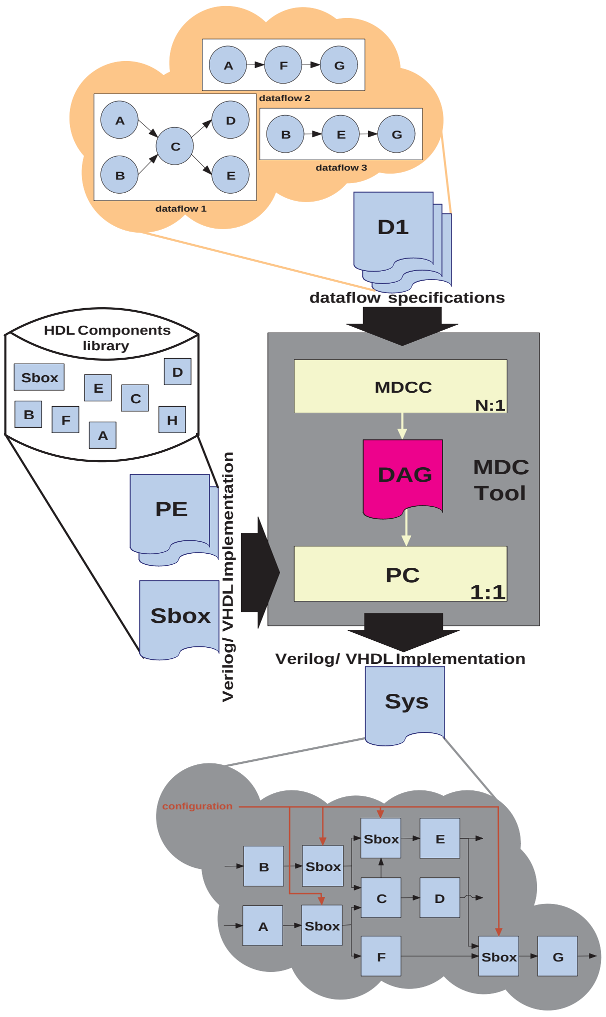


Figure 6.1: Coarse-grained reconfigurable platform composition flow.

platform within a single clock cycle and without any need of a reset phase. Obviously, to exploit at the most the MDC tool strength, the kernels should share the actors so that the resource saving on the final platform will compensate the additional hardware necessary to handle runtime reconfiguration.

Since the MDC tool is not a high-level hardware synthesizer, this step requires an HDL library of components, the FUs implementing the required actors and also the interconnection modules. Therefore, the HDL implementation of the actors must be manually coded by the user or defined using HDL generation tools [81] while the NL files which describe the single kernels could also be generated using the open source graphical Graphiti tool [25], greatly reducing the overall development time.

6.2.2 HDL components library and Communication Protocol

As previously announced, this approach needs to have the HDL implementation of the various actors instantiated in the reconfigurable platform due that the MDC tool is only able to manage the interconnections among them starting from the dataflow description of the single kernels. In order to deal with the IEEE 754-1985 single precision floating-point number representation [33], an appropriate HDL components library of standard-compliant FUs has been developed. It is built up of arithmetic and control flow modules managing numbers composed by the three fields: sign, mantissa and exponent. The arithmetic modules are an *adder*, a *multiplier*, an *absolute value calculator* and a *comparator*. The latter two are the simplest. The *absolute value calculator* just reverts the sign bit if it is set. The *comparator* is a mere integer comparator, since the represented rational numbers are ordered as the integer ones.

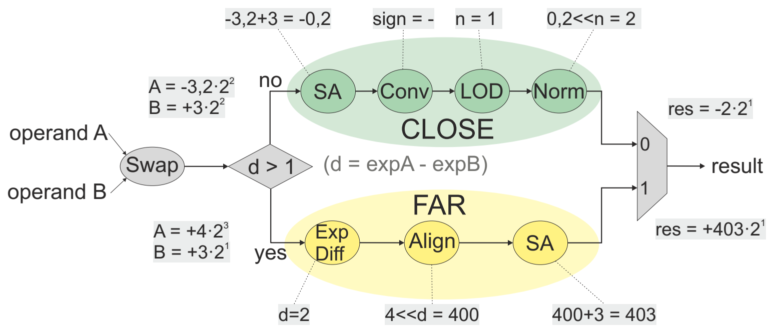


Figure 6.2: Floating Point Adder: the Block Diagram of the Two Path algorithm.

The multiplication operation is performed by the sum of the exponents and the product of the two *mantissa* fields, being careful to the biased exponent[1] and underflow/overflow situations. In the implemented *multiplier*, the product is computed in two clock cycles.

The most complex arithmetic block is the *adder*. Its implementation leverages on a variant of the Two Path algorithm [66], diversifying the computation according to the distance between the exponents of the two addends (Fig. 6.2). If the distance is less than/equal to 1 the *CLOSE* case is executed, otherwise the *FAR* case is. In the *CLOSE* case the two *mantissa* fields are summed using a *Sign Addiction* (*SA*). Through a detection process, called *Leading One Detection* (*LOD*), it is calculated the number of shifts necessary to normalize (*Norm*) the result[2]. In the *FAR* case, the *mantissa* fields are aligned (*Align*), according on the distance between the exponents (*Exp Diff*). The sign fields are added (*SA*). In both the FAR and the CLOSE cases, a preliminary (*Swap*) operation may be performed to switch the *adder* operands (the greatest one should always be the first). Three clock cycles are required to perform a floating point addition.

Beyond the arithmetic blocks, needed for the chosen application, also flow control blocks have been implemented in the library: two *demuxes*, respectively with 3 and 4 outputs, each one having an input buffer that stores a stream of incoming data and sends them towards multiple output channels at the same time; an *out_ filter* module to supervise, at the same time, two filtering processes useful to perform each stage of the denoising; finally, a *thresholder* which establishes whether the input data stream has to be forwarded or not.

All the FUs within the designed HDL components library obey to the communication protocol depicted in Fig. 6.3. It means that each FU has been encapsulated with an homogeneous wrapper managing the handshake and allowing the correct streaming of data in the system [67]. Dataflow paradigms typically implement a FIFO-based communication proto-

[1] In the IEEE 754-1985 single precision number representation a bias value of 127 is subtracted to the exponent.

[2] The integer part of the *mantissa* is represented always with only one implicit bit set to 1.

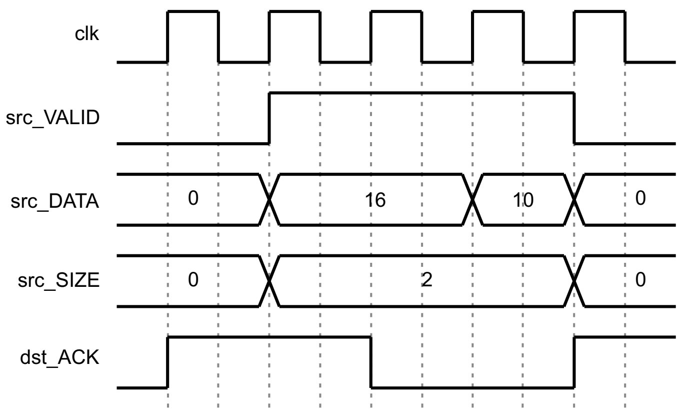


Figure 6.3: Multi-kernel datapath of the communication protocol.

col. To limit the required hardware resources, it has been adopted a single register for each I/O port of the FU instead of a FIFO memory allocated on each communication channel.

6.2.3 Computing kernels

To maximize the FUs reuse in order to obtain a low-power hardware platform with the minimum actors compliant with respect to the constraints imposed by the target application, it is fundamental to choose the single kernels with the higher computational complexity which use the same basic operations to perform the relative processing.

Figure 6.4 highlights the various kernels identified in the reference application based on the use of a template-matching spike sorting decomposed into the three typical parts (spike detection, alignment and sorting) plus a preliminary step of wavelet denoising, as detailed in the previous chapters, except the use of the final classifier stage. The resulting hardware implementations, obtained using the MDC tool, must be able to work on-line both when the action potentials of the single neurons are being identified determining the spike morphology with which each of them fires (training phase) and when they have been already identified recognizing, for each detected spike, what has been the neuron that has generated it (sorting phase).

Regarding to the wavelet denoising processing stage, considering an input sampling frequency of $12kHz$, the block diagram shown in Fig. 6.5 represents a possible solution, applying the non linear band-pass filtering between 375Hz and 6kHz, able to remove the low frequency interferences. In this scheme, it has been highlighted three different kernels: the single-stage decomposition (*dec*), the single-stage recomposition (*rec*) and the thresholding (*thr*).

The *dec* kernel (Fig. 6.6) performs a pair of FIR filtering operations (low pass and high pass) on the same input signal, generating the correspondent *approximation* and *detail* outputs. Looking at the Fig. 6.5, 4 successive executions of this kernel will be required to complete the decomposition phase, one for each level in the proposed solution. The one-input/three-output *demux* acquires an input stream containing the data sample and two filter coefficients at a time. The multiply and accumulate (MAC) operations need a pair of one *multiplier* and one *adder*, in parallel, for both the low pass filter and the high pass one. The *out_ filter* actor monitors the filter operations and gives at output the *approximation* and the *detail* samples related to the provided input data samples.

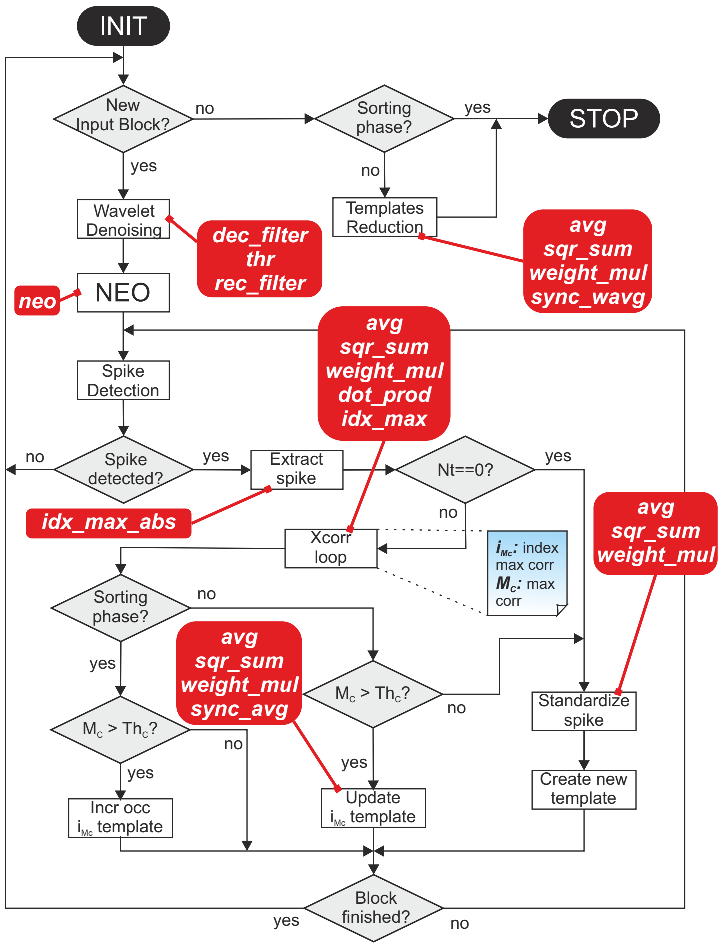


Figure 6.4: The flow diagram of the implemented algorithm.

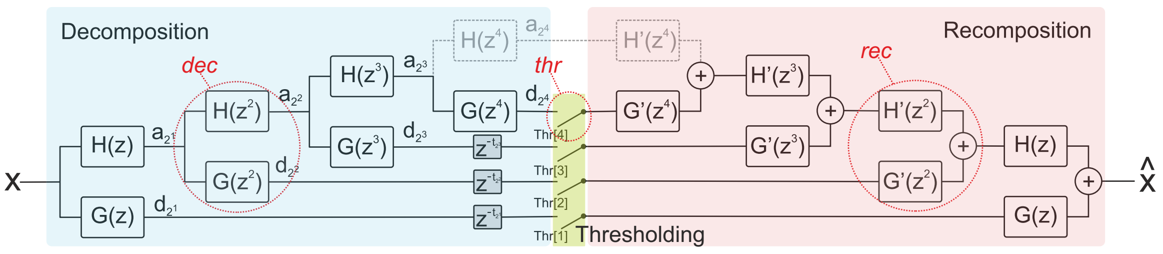


Figure 6.5: The proposed wavelet denoising solution block diagram with the three identified kernels

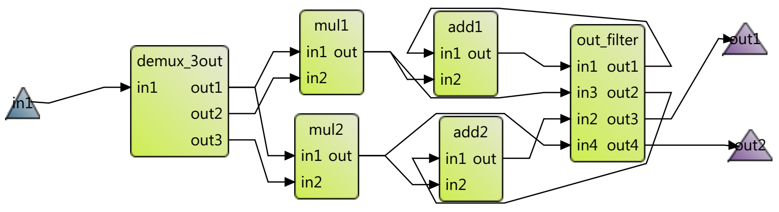


Figure 6.6: Single-stage decomposition dataflow kernel, *dec*.

The *rec* kernel (Fig. 6.7) is very similar to the *dec* one. As already said for the *dec* kernel, 4 successive kernel executions are required to perform the whole recomposition phase for the considered WD scheme. The initial *demux* provides a couple of input samples/relative coefficients both to the low pass and the high pass filters. At the end of the recomposition kernel flow an *adder* actor is required to correctly reconstruct the denoised signal, bringing back together the *approximation* signal and the *detail* one.

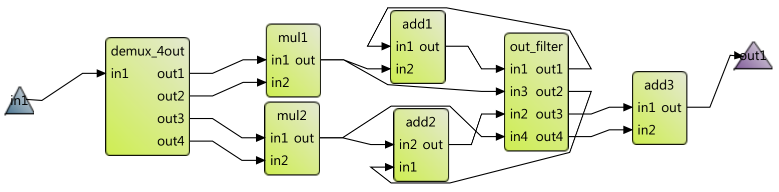


Figure 6.7: Single-stage recomposition dataflow kernel, *rec*.

The *thr* kernel (Fig. 6.8) compares the absolute value of an input sample with a threshold. If the sample (whose absolute value is extracted with an *absolute value calculator* actor) is greater than the threshold, it is forwarded to the output, otherwise the output is set to zero (a *thresholder* actor evaluates the *comparator* response and properly sets the output data).

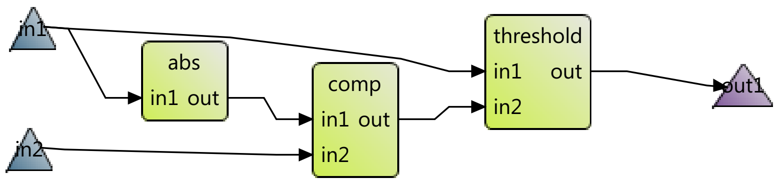


Figure 6.8: Thresholding dataflow kernel, *thr*.

In the same way, the remaining kernels have been implemented exploiting the actors belonging to the HDL components library. Thanks to the sequentiality of the operations, it is not necessary to instantiate other actors to execute, for instance, the arithmetic mean of a data vector (*avg*), the multiplication by a constant (*weight_mul*) or the squared sum (*sqr_sum*). Finally, the overall reconfigurable datapath includes only 3 adders, 3 multipliers, 1 subtractor, 1 comparator and 1 absolute value calculator to perform the various identified kernels.

6.3 FPGA test environment

To verify pros and cons of the proposed approach, a Xilinx FPGA Spartan-3E 1600 Development Board has been used as the target device on which it is possible to map the test environment system. The MDC-derived reconfigurable architecture has been integrated into a coprocessor at the disposal of a MicroBlaze soft-core processor to accelerate the execution of the single kernels. A block diagram of the whole test system is shown in the Fig. 6.9.

A point-to-point connection has been adopted between the coprocessor and the MicroBlaze to reduce the communication latency. The Fast Simplex Links (FSLs), composed of 32-bit wide uni-directional channels, have been instantiated on the system. The *FIFO depth* of the FSL-based links have been fixed to 32, a good design compromise to reduce the FPGA slices utilization and, at the same time, to limit the number of "full" conditions.

The runtime coprocessor configuration, based on the MicroBlaze requests for the execution of the single kernels, needs the preliminary sending of a control word defining the value

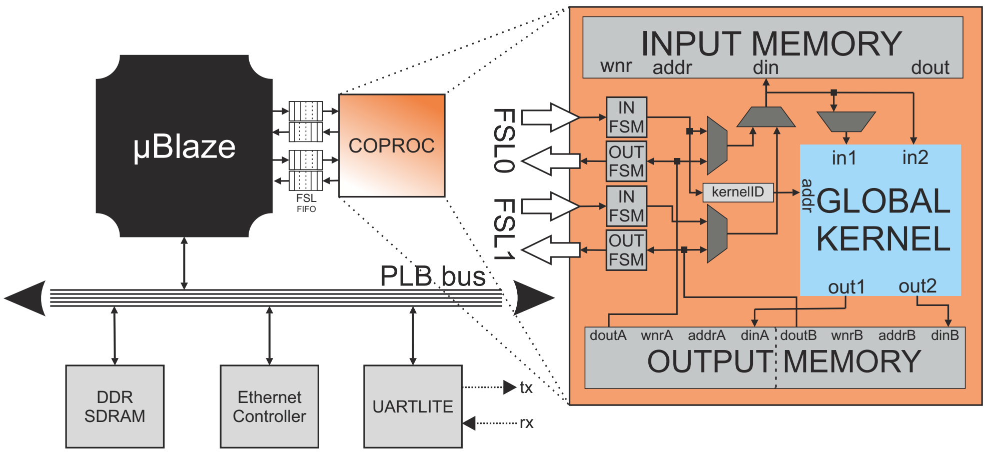


Figure 6.9: Test system block diagram.

of the possible parameters. Among them, it is necessary to specify the *Kernel ID* which represents the correct index of the invoked kernel to be computed by the *Global Kernel* block, properly configuring the Sboxes. For instance regarding to the wavelet denoising, the designer can choose:

- the number of levels used in decomposition/recomposition;
- the number and the values of the coefficients used for the filters based on the selected mother wavelet;
- the possibility of removing the last *approximation* signal determining an equivalent band-pass non-linear filtering;
- the threshold values for the *details* at each decomposition level;
- the b_len number of samples for each input frame (considering 512 as the maximum possible value for the memory resources available on the selected FPGA).

The *Input Memory* block has been instantiated to temporarily store the input data which must be used in the following invocations of the coprocessor. For example, during the execution of the wavelet denoising stage, at the first call of the *dec* and the *rec* kernels the processor transmits to the coprocessor the filters coefficients which are saved in this memory in order to be reused by the subsequent levels of the filtering trellis. Similarly in the decomposition phase, the processor sends to the coprocessor also the current input buffer that is going to be stored in the *Input Memory* block. Storage is necessary since each N-tap filter output depends on the previous $N-1$ old input samples.

Processor to coprocessor handshake and storage on the *Input Memory* block are managed by the *Input FSM* blocks, which are in charge also of the proper address generation for the *Input Memory* access. The *Input Memory* block contains the $H(z)$ and the $G(z)$ filter coefficients along with the input samples in the *dec* kernel case, and the $H'(z)$ and the $G'(z)$ filter coefficients along with their respective input samples in the *rec* kernel case. The latter situation imposes the memory size since two buffers of samples have to be stored.

Global Kernel output data are stored into the *Output Memory* block. Local storage is convenient to facilitate both the decomposition, just from the second level on, and the recompostion phases whereas in sorting phase, it can contain the various samples of the obtained templates with respect to verify the similarity with the incoming detected spikes. This allows to reduce the processor-coprocessor communication overhead.

The *Output Memory* block is able to perform two simultaneous read/write operations in the same system clock cycle. In fact, internally this memory is driven by a clock signal two times faster than the system one. Moreover, a dual-port memory is adopted to store the output of the $H(z)$ and the $G(z)$ filters when a *dec* kernel is executed. At the end of the decomposition phase, when the entire sequence of *dec* kernels has been sequentially executed, the *Output Memory* contains the various *detail* signals $a_{2^i}[n]$ and the last *approximation* signal $d_{2^{N_{lev}}}[n]$ adopting different base addresses. Coprocessor to processor handshake and storage on the *Output Memory* block are managed by the *Output FSM* block.

The assembled testing environment allows the user transmitting the input data and the coprocessor parameters to the MicroBlaze at runtime. To this aim, a Xilinx Ethernet controller, accessible by the MicroBlaze, has been connected to the PLB (Peripheral Local Bus) exploiting an UDP communication between the FPGA and a host PC. For this reason, even in this case the LightWeight Internet Protocol (LwIP) software library, which allows to implement the TCP/IP protocols stack optimized for embedded systems mapped on FPGAs, has been used. The related source code, the MicroBlaze application and the temporary data buffers are linked to the external DDR SDRAM attached to the same PLB.

6.4 Experimental results

To evaluate the benefits of this proposed approach, it is necessary to verify the system functionality of the reconfigurable coprocessor at the disposal of the MicroBlaze to perform the various processing steps of the considered neural decoding algorithm. To this aim, it has been used the same datasets of simulated neural signals used in the previous chapters, constructed starting with a database of physiological spikes, representing the action potentials obtained during real experimental recordings [72] on animals at the CNS level. To simulate the background noise, spikes from different neuronal cells at random times and amplitudes were overlapped to the relevant neural activity, considering that the useful signal bandwidth for these signals, sampled at the frequency of $12kHz$, is between $300Hz$ and $3kHz$.

6.4.1 Accuracy results

The first thing which must be tested is the presence of action potentials at the output of the WD, removing all the background noise without corrupting the signal of interest (i.e. avoiding to filter out any relevant spikes). As said before, the significant bandwidth of the exploited test data lies between $300Hz$ and $3kHz$, so the wavelet denoising stage has been customized to obtain an equivalent bandwidth between $375Hz$ and $3kHz$ (4 levels of decomposition/recomposition), discarding the last approximation signal and the lowest-frequency detail using the simpler *Haar* family as mother wavelet which determines the value of the filters coefficients.

Figure 6.10 shows the input signal (top), corrupted with a low level of background noise, and the denoised output signal (bottom) obtained adopting the proposed reconfigurable hardware solution using frames of 500 samples and the *Haar* wavelet. The mother wavelet influences the number of taps, in this case 2, of the $H(z)$ and $G(z)$ in decomposition phase and of the $H'(z)$ and $G'(z)$ in recomposition phase, and those of the descending filters. As can be noticed, the background noise has been properly removed, reaching a good degree of accuracy.

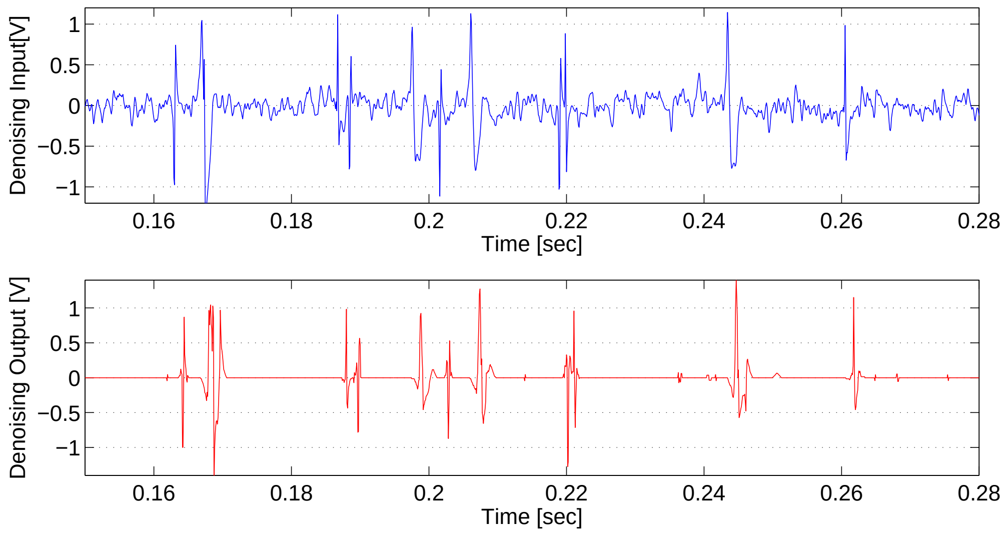


Figure 6.10: Input (top) and output denoised (bottom) signals. Low level background noise. *Haar* mother wavelet.

Filtering performance has been assessed also considering a larger background noise overlapped to the input signal. Figure 6.11 shows the input signal (top), corrupted with a larger background noise, and the denoised output signal (bottom). As can be noticed, in the denoised output signal the action potentials are again clearly visible, demonstrating that even a larger background noise is efficiently filtered out.

The reconfigurability of the proposed approach allows selecting different mother wavelets without any modification of the system based on the current characteristics of the processed signal. Different filter coefficients have to be transmitted to the coprocessor, whose overall structure does not change. Opting for a *Daubechies 2* mother wavelet modifies the number of filters taps, incremented to 4, leading to the results shown in Fig. 6.12.

To verify the performance of the hardware implementation even at output of the template-matching spike sorting, among the possible test signals, the *Easy1* one has been selected since it is the only one where the spike waveforms from different neurons present a correlation value lower than 0.9 between them. This is fundamental taking into account that the considered spike sorting algorithm must be able to recognize the differences among the various spikes, associating each of them to a particular template characterized by having the largest correlation with it. *Easy1* signal includes the activity of three neurons and, in 60 seconds, each neuron fires approximately 1100 times. Table 6.1 shows the results in terms of number of spikes associated to the different templates autonomously identified by the algorithm, limitedly to the $N = 10$ most significant ones. Even increasing the background noise, the proposed system is able to detect the spikes with a reasonable accuracy.

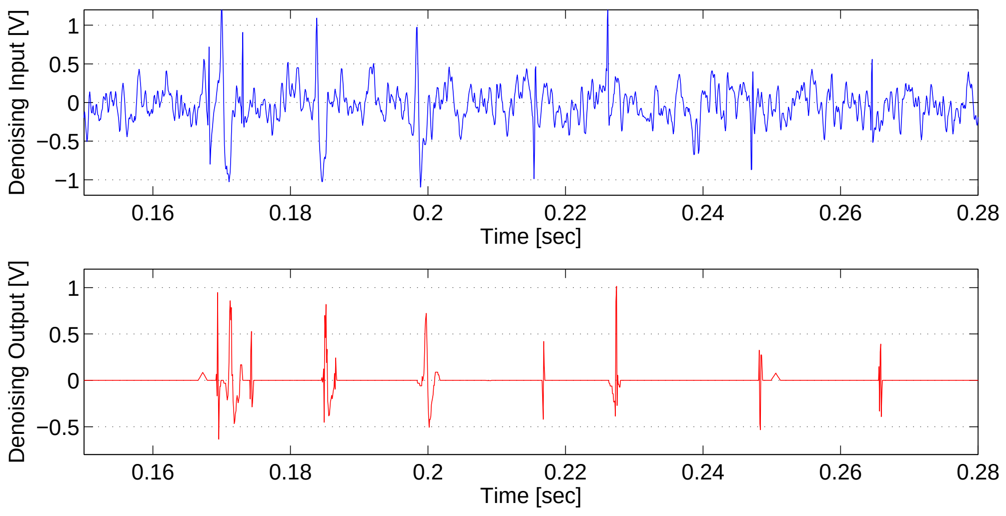


Figure 6.11: Input (top) and output denoised (bottom) signals. High level background noise. *Haar* mother wavelet.

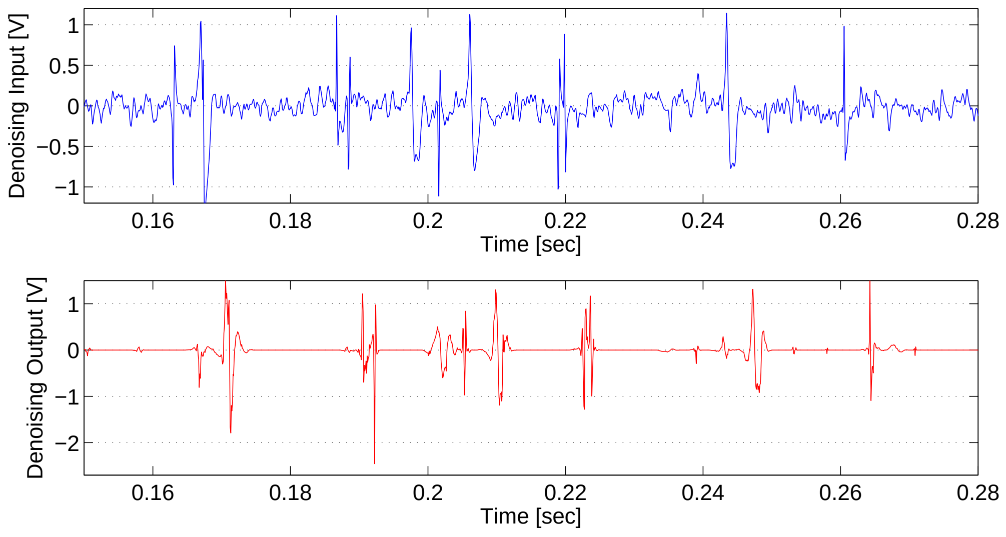


Figure 6.12: Input (top) and output denoised (bottom) signals. Low level background noise. *Debauchies 2* mother wavelet.

6.4.2 Latency analysis

Even though the main goal of the proposed approach is that of semi-automatically creating small/low-power architectures implementing complex dataflows fostering FUs reuse, the hardware implementation can mark significant improvements also in terms of latency, compared to a purely software implementation, in order to satisfy the relative real-time constraints of the application under consideration. A comparison in terms of execution cycles of the individual kernels, performed exploiting the MicroBlaze processor alone or the architecture including also the reconfigurable coprocessor, is presented in Fig. 6.13. The better performance with the coprocessor represents a lower bound, since the minimum number

Table 6.1: Number of spikes associated to each template while varying the background noise level of the synthetic signal.

Noise	Template 1	Template 2	Template 3	Template 4	Template 5	Template 6	Template 7	Template 8	Template 9	Template 10
0.05	1080	1016	992	25	22	11	10	9	6	5
0.10	1120	1002	998	130	46	17	9	8	7	7
0.15	1079	1039	979	246	11	9	6	5	1	1
0.20	1010	788	723	469	12	6	4	4	3	3
0.25	1065	870	868	60	12	3	3	0	0	0
0.30	1053	930	582	67	0	0	0	0	0	0
0.35	1124	854	593	83	0	0	0	0	0	0
0.40	742	561	262	45	0	0	0	0	0	0

of FUs has been instantiated in order to save both area and power. Giving up some saved resources, the performance can be improved in the light of a defined number of channels and real-time constraints.

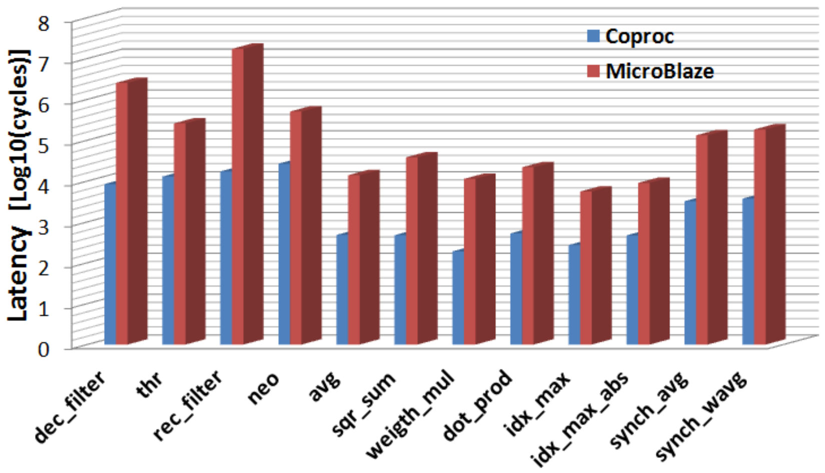


Figure 6.13: Latency comparison between a totally software solution by the MicroBlaze core and the architecture including also the low-power reconfigurable coprocessor.

6.4.3 Area occupancy and power consumption

Synthesis have been performed to evaluate the effectiveness of the proposed approach in terms of area and power saving. It has been done a preliminary study on the reconfigurable datapath generated by the MDC tool only considering the single kernels to perform the pre-processing wavelet denoising stage and then all the single kernels identified in the flow diagram of the algorithm shown in Fig. 6.4. Initially, it has been considered as target device for the *Global Kernel* the same FPGA development board adopted for the test environment

Table 6.2: Number of adder and multiplier FUs composing the considered wavelet denoising implementations.

IMPL	adders	multipliers
MGK	3	2
SGK	6	4
Parallel WD	20	32

exploiting the dedicated Xilinx Synthesis Technology (XST) tool. Afterwards, it has been performed a second synthesis trial with RTL Compiler of the Cadence SoC Encounter commercial release, using an ASIC 90nm low-power CMOS technology.

Regarding to the wavelet denoising, the *Global Kernel* assembled with the MDC tool (hereafter named *MGK*) has been compared with:

- the datapath obtained by the 3 identified kernels (*dec*, *rec*, *thr*) without any resource sharing (Static Global Kernel, *SGK*);
- the parallel system (*Cascaded Kernel*) implementing the wavelet denoising solution in Fig. 6.5.

Results reported for the *Cascaded Kernel* constitute an estimation, whose values are determined taking into account the number of required FIR filters. The chosen FIR filter implementation model is the Direct Form I [4], composed by two multipliers and an adder. Such estimations leverage on the results achieved for the implemented floating-point HDL components library, whose synthesis outcomes for the *multiplier* and the *adder* FUs are reported in Tab. 6.3. It has been enforced in XST the use of MULT18X18SIO hardware multipliers, avoiding the less efficient and more area-hungry implementation on distributed logic. For these estimations, it has been decided to discard the contributions of all the other components (i.e. those used in the thresholding phase) that are less significant in terms of area and power.

Table 6.2 summarizes the amount of resources used for the three aforementioned *Global Kernel* implementations. For the *Cascaded Kernel* it has been estimated the usage of 32 multipliers and 20 adders, a decomposition phase and a recomposition one has been considered, both composed of four levels each containing in turn two FIR filters ($2 \times 4 \times 2 \times 2 = 32$ multipliers and $2 \times 4 \times 2 \times 1 = 16$ adders) plus an additional adder at the end of each recomposition level ($4 \times 1 = 4$ adders).

The synthesis results for the FPGA target device are exposed in Tab. 6.4. It is evident that adopting the MDC tool has a significant impact on the resource usage. The *MDC Global Kernel* in terms of slices is able to achieve 36% of saving compared to the *SGK* and 82% compared to the *Cascaded Kernel* estimation. Considering the Xilinx multiplier primitives, the resource saving percentage is 50% and 97% respectively. The MDC-based solution presents a very small resource occupation percentage on the target device, whereas the *Cascaded Kernel* requires a considerable amount of resources and, due to its large multipliers request, cannot be mapped onto the considered FPGA.

Tab. 6.5 summarizes the results of the FPGA synthesis for the unique dataflow implementation as generated by the MDC tool (*MGK*) in comparison to those achievable for the architecture resulting from the implementation of all the kernels of the whole decoding algorithm but without resource sharing (*SGK*) and the whole *Coprocessor* (including the MGK

Table 6.3: Synthesis results for the most interesting FUs.

FPGA			
FU	**slices [%]**	**FFs [%]**[a]	**MULs [%]**[b]
adder	341	116	0
multiplier	138	131	11
ASIC			
FU	**area** [μm^2]	**power** [mW]	**freq** [MHz]
adder	8644	1,679	555,56
multiplier	113497	2,031	357,14

[a]Flip Flop slices.
[b]MULT18X18SIO dedicated multipliers.

Table 6.4: FPGA synthesis results only considering the kernels for the wavelet denoising

IMPL	**slices [%]**	**FFs [%]**[a]	**MULs [%]**[b]
MGK	14	5	22
SGK	22	8	44
Cascaded Kernel	76	22	356

[a]Flip Flop slices.
[b]MULT18X18SIO dedicated multipliers.

Table 6.5: FPGA synthesis results considering all the kernels of the algorithm.

IMPL	**slices [%]**	**FFs [%]**[a]	**MULs [%]**[b]
SGK	81	23	100
MGK	26	7	33
Coprocessor	32	9	36

[a]Flip Flop slices.
[b]MULT18X18SIO dedicated multipliers.

wrapped by an external layer that allows its easy exploitation). As it can be seen, the reconfigurable MGK allows saving about the 61% of the slices available on the chosen device compared to the SGK.

About the ASIC synthesis, Tab. 6.6 reports the results extracted using RTL compiler. It is confirmed, both considering the area occupancy (the synthesis reports provide the effective system area on a die) and the power consumption (the synthesis reports provide an estimation of the consumption), that the proposed MDC-based approach is able to provide superior performance. The area estimation of the *MDC Global Kernel* is respectively 40% and 86% smaller than the *SGK* and the *Cascaded Kernel* ones. The power consumption estimation of the *MGK* is 40% less than the *SGK* one, whereas it is the 90% less than the *Cascaded Kernel* one. The maximum operating frequency is fixed by the floating point multiplier, so that it is equal for all the considered *Global Kernel* implementations. This result also for the *MGK* demonstrates that, for this particular application, the overhead of introducing the *Sboxes* to handle reconfigurability does not impact on the frequency. Consequently, even though there may be chains of *Sboxes* introduced by the merging kernel process, these do not contribute

Table 6.6: ASIC synthesis results.

IMPL	**area** [μm^2]	**power** [mW]	**freq** [MHz]
MGK	86353	12,880	357,14
SGK	142927	21,499	357,14
Cascaded Kernel	604784	128,593	—

Table 6.7: ASIC synthesis results.

IMPL	**area** [μm^2]	**power** [mW]	**freq** [MHz]
SGK	462807	61,556	357,71
MGK	133949	14,691	312,5
Coprocessor	159325	15,160	208,33

to the critical path. In the *Cascaded Kernel* case the value of the operating frequency is not reported since an effective implementation of the system has not been performed.

Finally, the ASIC synthesis results considering all the kernels identified in the decoding algorithm are presented in Tab. 6.7. The MGK still has significant advantages compared to the SGK, saving 71% of the area and 76% of power dissipation. In this case, it has been obtained a 13% decrease of the maximum operating frequency. Compared to the MGK, the coprocessor requires a larger area (+19%) and dissipates more power (+3%). It suffers further penalties in terms of the operating frequency (-42%) compared to the SGK.

Summarizing, both the synthesis trials, targeting FPGA and ASIC devices, have showed that the adoption of the MDC approach in the implementation of the on-line neural decoding algorithm for neuroprosthetic applications is a very effective choice, especially when it is necessary to develop an implantable/portable solutions. In this way, the implemented solution, but more generally the proposed approach itself, can be very useful where there are stringent physical constraints, mainly related to area and power. The performance in terms of latency can also be improved by tuning the implemented parallelism in the light of a defined number of channels and real-time constraints, by using more than one reconfigurable global kernel in order that they can be exploited to perform the same or different kernels at the same time in a parallel way, due to the fact that each one can execute the relative processing only in a sequential way.

Chapter 7

An FPGA-based MPSoC for On-line Neural Signal Decoding

7.1 Introduction

Opting for a coarse-grained reconfigurable approach to automatically create a low-power implementation of the target decoding algorithm with the minimum hardware resources determines some difficulties in the fulfilment of the real-time constraints imposed by the application under consideration. It has been demonstrated that the use of a reconfigurable coprocessor allows to accelerate some computational intensive kernels in comparison to a totally micro-programmed case but the desired processing performed by the Global Kernel and invoked by the MicroBlaze processor can be done only in a sequential way, once a time. For this reason, in the perspective of the development of a multi-channel decoding implementation which must satisfy the tight real-time constraints, it is necessary to exploit some effective techniques of parallelism even considering the problems related to the power consumption to allow portability of the resulting solution.

As detailed in the previous chapter, spike sorting algorithms, especially those based on template-matching approaches, are complex considering the relatively high sampling frequency required by the neural signals [18] in comparison to other physiological signals (e.g. EEG) and the possible presence of multiple channels [65]. This hampers the difficulty of the design of low-power miniaturized embedded systems that could be implanted enabling low-bandwidth communication with the external prosthesis, limited to the motor commands. The aim of this part of the research activity is to make a step further in the definition of novel power-efficient methodologies for the implementation of such algorithms. The main idea is to take profit from the inherent parallelism in the reference application to effectively execute it on a custom Multi-Processor System-on-Chip (MPSoC). For this reason, the same PNS signal decoding algorithm considered in the previous chapters has been taken into account [16] thanks to the fact that its on-line version has revealed good performance even in case of low SNR conditions, both on a synthetic dataset of neural signals and a database of real recordings extracted during in-vivo experimental tests on sedated animals.

As described before, it presents two main critical features that must be duly taken into account. Firstly, the application has to be executed in real-time to be safely capable of decoding every useful information in a timely manner for the control of the neuroprosthesis, providing a robust support to the patient requiring significant processing power. The idea is to exploit the potentiality of the MPSoCs, increasing computing capabilities through parallel computing but using lower working frequency to reduce the power consumption. Secondly, spikes distribution over time is not regular. Due to the relationship between spikes distribution and the motion intention, the workload that the processing system has to undergo is deeply dependent on it and, in turn, on the signal itself. The performance levels needed in each different timing interval are hardly predictable and can be very changeable. To ensure robustness, worst-case conditions must be assumed. Such assumption results in an over-dimensioning of the system that has to be counter-balanced with adequate power management measures to improve battery lifetime.

To this aim, it will be proposed an approach that is focused on the software-based control of the power consumption. When, due to the absence of useful information in the input stream, the processing elements in charge of analysing the spikes extracted from the signal are inactive, they are put in low-power mode by calling an adequate Application Programming Interface (API) in the application code. The demonstration of the proposed approach is provided implementing it on a low-cost FPGA. The designed MPSoC is capable of processing the neural signal in real-time when clocked at a reasonable frequency. The power-reduction API selectively gates the clock signals routed to those processors that are active only when the presence of useful information is detected. In such a proof-of-concept system, the power reduction is thus limited only to the dynamic part of the power dissipation which usually represents the largest contribution in terms of energy consumption. However, it demonstrates the possibility of applying more aggressive power reduction techniques on a prospective ASIC implementation of the system which can be placed near the stump of the prosthesis working with batteries as power source allowing portability for the resulting solution.

7.2 Related work

As already mentioned, spike sorting techniques aim at recognizing the different neurons, whose firing activity has been recorded, on the basis of the information encoded at different levels in the morphology of their action potentials [46]. Even when selectivity of the used multi-channel electrodes is high, the activity of multiple motor neurons can be found on the same channel, thus requiring the adoption of such techniques in order to extract the motor commands destined to control different muscles. Depending on the chosen neural interface, the number of channels can considerably change from 4-8 for PNS intraneural and epineural electrodes to more than one hundred for microelectrodes arrays in cortical implants. Considering the relatively high bandwidth required, typically around $5-6kHz$, and the invasiveness of the electrodes, the best solution would be that of implanting the whole decoding platform coming out only with the control signals for the prosthesis, requiring far less bandwidth. However, the real-time constraints required by an usable device and the low-power profile aimed at a long battery life and, more, at a reduced heating of the patient's tissues are contrasting.

Several works on neural signal processing have been presented in the years, the largest part of them proposing FPGAs [92, 89, 8] as implementation target to guarantee more flexibility than ASICs [15, 71] with more parallelism than general-purpose processors. In [92], a fully implantable programmable neuroprocessor mappable on a low-power nano-FPGA is presented. It manages data acquisition and reduction by particular compression techniques in order to minimize the output bitrate exploiting the sparse representation of the neural signals. In this way, it is possible to overcome the limitation of the wireless telemetry bandwidth by transmitting only the samples associated to the detected spikes to an external device for cortically-controlled Brain-Machine Interfaces. This solution has been tested on raw extracellular signals recorded through micro-electrode arrays chronically implanted in the brain of sedated rats. The feasibility of this approach in terms of power consumption has been investigated on standard CMOS VLSI [94]. However, in this approach, the computational complexity is shifted at downstream of the implantable device in order to perform the decoding which can be executed on many-core platforms [14] or FPGA-accelerated solutions [24].

Other energy-efficient implementations for multi-channel spike sorting have been published so far [37], most of them concerning the processing of signals coming from the CNS. Some of them focus on the analysis, in terms of necessary hardware resources and accuracy, of some typical processing steps of spike sorting algorithms [61, 23], optimized in order to be mapped on a dedicated chip. In these cases, massive parallelization is in contrast with the low-power requirements. Despite the approach aiming to solve the neural signal decoding trough the PNS seems to be the most attractive for the time being [56], there is a lack of studies in terms of architectures able to cope with the application constraints. As demonstrated in the previous chapters, the same algorithm has been ported on a complex VLIW floating-point processor by the Texas Instruments but the claimed real-time results have been obtained on a device clocked at $300MHz$: such an architecture, and the operating frequency, determines an excessive contribution in terms of dynamic power consumption that is not allowable in case of implantable or portable solutions. The methodological aspects related to power-efficient and effective multi-processor architectures aimed at implementing in real-time state-of-the-art neural signal decoding algorithms seems to lack in the scientific literature, and will be preliminarily addressed in the next sections.

7.3 Target application and related constraints

Even though the target algorithm was originally conceived for single-channel PNS signal decoding, this choice does not limit in principle the prospective scalability of the proposed approach. Without considering the final SVM classifier stage, as in the previous coarse-grained case, which is not the most computational intensive processing step of the algorithm, it takes as input a single PNS signal channel whereas its output is an indication, for each detected spike, of the class the spike belongs to. Classes are represented by an average spike waveform (hereafter called *template*) univocally associated to a single motor neuron, so that every spike resembling a given template can be considered as produced by that neuron. The results of such a processing could be used to train a classifier to recognize the movement intentions, looking at the global firing activity in a sliding window of a predefined size.

Such a process could be performed in a remote computing facility with more relaxed requirements in terms of low-power consumption sending out, through a wireless network interface, the class identifier associated to the last processed spike. It will be considered in this case only the part of the algorithm which must be performed in on-line processing mode, i.e. when all the templates have been already created, due that it can be done at off-line by using for example an equivalent Matlab implementation as specified in the literature

[76]. Fig.7.1 shows the flow diagram of the reference decoding application, highlighting the presence of the various steps involved in the on-line processing which are described in detail in the following subsections.

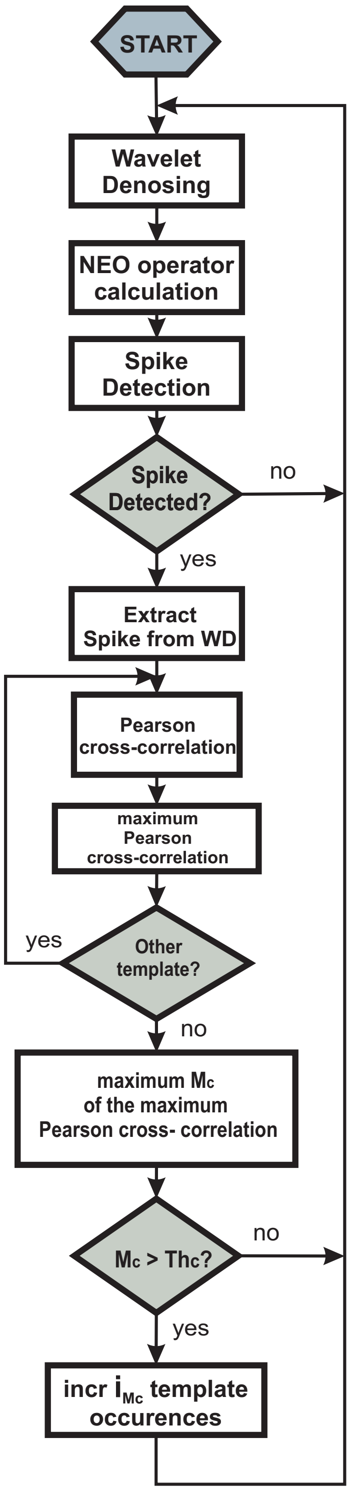

START
Wavelet Denosing
NEO operator calculation
Spike Detection
Spike Detected?
no
yes
Extract Spike from WD
Pearson cross-correlation
maximum Pearson cross-correlation
yes
Other template?
no
maximum Mc of the maximum Pearson cross- correlation
no
Mc > Thc?
yes
incr i_Mc template occurences

Figure 7.1: Flow diagram of the target neural signal processing algorithm (Thc represents the cross-correlation threshold).

7.3.1 Wavelet Denoising

As described before, *Wavelet Denoising* is typically used to remove noise that lies in the same bandwidth of the signal, especially when it can be approximated as a Gaussian distributed random source. Compared to traditional linear filtering it could be a very useful tool for low-SNR signals. According to what is shown in Fig. 7.2, the same settings of the previous DSP implementation have been used:

- the so called *à trous* algorithm [80] to allow time invariance and the same resolutions at the various decomposition bands;
- an *Hard* approach for the thresholding phase applied to the detail signals (the coefficients with values below the threshold are discarded whereas those above the threshold are retained);
- the cost-effective *Haar* family as mother wavelet, which determines the coefficients values of the FIR filters at decomposition and recomposition phase;
- an input sampling frequency f_s of $12kHz$, using 4 trellis levels and clearing the last approximation signal in order to perform a band-pass non-linear filtering in the range between $375-6000Hz$ considered as significant output bandwidth.

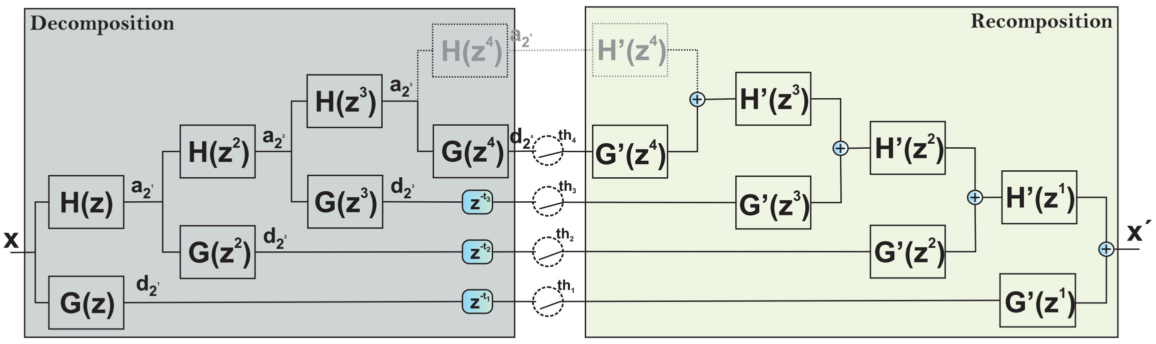


Figure 7.2: The Wavelet Denoising scheme.

Filtering operations are convolutions requiring several multiply-and-accumulate operations. Compared to the block-based processing proposed in the previous DSP implementation, introduced to improve the relative latency performance, a sample-by-sample approach has been exploited in order to limit the size of the data structures to be stored in the local memory of the assigned processor. The WD buffering required by the filtering operations is performed locally, without impacting on functional and timing performance.

7.3.2 Spike Detection

The neural activity represented by the spikes should be detected in the denoised signal in order to trigger the spike sorting phase. The detection mechanism is based on an amplitude threshold derived by applying the Non-Linear Energy Operator (NEO) on the WD output. Such operator has the advantage of emphasizing the presence of the pulses [36] since it assumes high values in correspondence of those sample windows that are characterized by high power and high frequency components, as typical of actions potentials. After the *NEO calculation* has been performed, a *spike extraction* procedure identifies the windows in the NEO signal for which the NEO values are above the threshold, storing the correspondent WD-processed samples in a buffer to be passed to the spike sorting phase. A head and a tail of ten samples before and after these events are also considered as part of the spike.

7.3.3 Spike Sorting

As shown in Fig. 7.1, the algorithm involves a template-matching spike sorting to obtain the class representing the motor neuron who fired the given detected spike. Morphological similarity between the current detected pulse and the set of reference templates, created off-line and representative of the neural activity recorded during the training phase of the algorithm, is evaluated through normalized cross-correlations based on the Pearson product-moment correlation coefficient. The number of reference templates (hereafter referred to as N_t) is a parameter of the algorithm, fixed off-line, descending from the training phase. It means that such a value can change between different executions as far as a new training is performed. This could be required due to a relative movement between electrode and nerve or to the progressive reaction to the living tissue to the synthetic material leading to a decreasing quality of the acquired signal [63].

In this particular implementation, both the maximum number of samples in a spike and in a template (Len) have been set to 40 since, at a sampling rate of 12 kHz, the window size of 3.3 ms is compatible with the size of isolated spikes. Cross-correlation has been preferred to simple correlation between spike and template in order to evaluate the best alignment. To this aim, the spike under analysis is centred in a buffer of $2 \times Len = 80$ elements on its maximum. Then, the algorithm calculates a Pearson product-moment correlation coefficient for each different overlap of Len samples between the buffer and the template, and the maximum value is taken as result. The normalized correlation method produces a result whose absolute value is ≤ 1. In order to speed up the computation, the templates could be loaded already standardized (i.e. centred and with unitary variance). If the overlapping part of the $2 \times Len$ buffer is also standardized (which must be performed at on-line computing ahead the mean and the standard deviation of that buffer part), the calculation of the Pearson coefficient is merely a dot product. For every overlap, the *spike standardization* takes place only once whereas the actual Pearson coefficient calculation, hereafter called *SpikeVSTemplate xcorr measurement*, has to be repeated for every template. The index of the template with the highest value is the final result of the spike sorting.

7.3.4 Real-time constraints

As already stated, the sampling frequency considered during the development of the system is 12 kHz. It has been assumed, quite conservatively according to in-vivo experiments, to need enough processing power to analyse one spike every 23 samples. This means that the worst case condition involves the presence of about 524 spikes/sec, defining a lower limit to the throughput of the entire system.

7.4 Parallelization and programming model

The considered neural signal decoding algorithm is a typical streaming application. The data received in input has to go through several processing steps, represented by explicit function calls, operating on a block of samples. The execution is thus the body of an infinite *while()* loop that triggers a new iteration every time that a new block of samples is available. The algorithm was studied and defined in the form of a sequential Matlab program, then converted in a C code as presented as presented in the Listing 7.1.

Listing 7.1: Sequential form of the target application

```
void main() {
  float  input_sample, denoised_sample;
  float detected_spike[SPIKE_SIZE];
  bool spike_detected;


  while (1){
    get_sample_from_ADC(input_sample);
    wavelet_denoising(input_sample, denoised_sample);
    spike_detection(denoised_sample, spike_detected, detected_spike);
    if (spike_detected)
          spike_sorting(detected_spike, output_class);
  }
}
```

Specifying the application as a sequential program is comfortable during the algorithm definition, since such form represents the most intuitive description, but it does not match the need to exploit parallelism. To this aim, it has been modified the code implementing the same algorithm using a Model of Computation (MoC) based on process networks. Such MoC is basically derived from Kahn Process Networks [35] and it is based on parallel processes that communicate through FIFOs. Each process is a repeated execution of a functional actor which receives input data from one (set of) FIFO(s) and writes output data to another one.

Process networks are well known to still represent a very intuitive way of specifying the application behaviour. The identification of the parallel tasks is almost straightforward, since each parallel node is a processing step of the functional process. Generally, each parallel node is in a form such as the Listing 7.2.

Listing 7.2: Example of the typical KPN structure code

```
  while (1){
    read(upstream_FIFO_1);
        ...
    read(upstream_FIFO_n);

    compute( );

    write(downstream_FIFO_1);
        ...
    write(downstream_FIFO_n);
  }
```

The nodes communicate with each other exchanging *tokens* through FIFOs. At each node iteration, a *compute* function elaborates the received tokens and produces new ones that have to be sent to the downstream nodes. The *compute* function represents the computation workload of the parallel node.

Theoretically the FIFO size in KPNs is unbounded, preventing an implementation on real memory-limited computing architectures. However, implementation is still possible [70, 22] by means of blocking reads and blocking writes to/from the FIFO, meaning that the process stalls when the input FIFO is empty or when the output FIFO is full. Using this kind of implementation, in general, can potentially bring to deadlocks if the chosen size of the FIFOs is not sufficient or if the scheduling of the processes on the different processors is not carefully tuned. In the considered case, such tuning steps are simplified by the very simple KPN graph resulting from the target application.

The Listing 7.3 represents a possible solution for the parallel node implementing the spike detection. The input token is the denoised block of samples, received from the node implementing the wavelet denoising. The *compute* function, *spike detection* in this case, elaborates it and, if a spike is detected, produces the detected spike as output token.

Listing 7.3: The parallel application node implementing the spike detection elaboration step

```
void parallel_spike_detection() {
  float denoised_sample;
  float detected_spike[SPIKE_SIZE];
    bool spike_detected;

  while (1){
    read(upstream_FIFO, denoised_block);
    spike_detection(denoised_block, spike_detected, detected_spike);
    if (spike_detected)
          write(downstream_FIFO, detected_spike, output_class);
  }
}
```

Once the application has been converted in a network of parallel nodes, the mapping of the different tasks on the different available processing elements has to be defined.

7.5 Sequential application profiling

The aim is to speed up the execution implementing a software pipeline. Provided that enough buffering resources are instantiated between different processes, the elaboration step envisioned in each process step can run in parallel on subsequent blocks of samples. The eventual performance can be pre-estimated using an analytical model of the dependence of the throughput on the node workloads. The work in [55] presents an approach for modelling the overall throughput of a KPN network, by calculating the throughput τ_{P_i} of every KPN process and propagating the minimum process throughput to the sink process (i.e. the output node). Each process P_i of the KPN can be annotated with a workload number W_{P_i}:

$$W_{P_i} = C^{P_i} + x \cdot C^{R_d} + y \cdot C^{W_r},$$

where C^{P_i} denotes the number of time units (i.e. clock cycles) required to execute the process function once, x and y denote how many FIFOs are read and written per process firing, and C^{R_d} and C^{W_r} denote the communication costs. The throughput of each process is, hence, $\tau_{P_i} = \frac{1}{W_{P_i}}$. The overall KPN throughput is denoted by τ_{out} and is defined as the average number of tokens produced by the network per time unit. Because it is known from [55] that the slowest process determines the system throughput, then $\tau_{out} = \tau_{P_{slowest}}$.

To find an effective mapping solution for the decoding algorithm, the pipeline stages have to be as balanced as possible. In order to provide this balance, an optimal partitioning of the application can require the splitting of some nodes. Moreover, the designer does not always have enough processing resources to map every process onto a different processing element. In this case, the process have to be clustered, identifying a "mapping" that minimizes the number of processors while keeping the same throughput. Thus, the choice of the optimal system configuration requires a preliminary profiling phase. The evaluation of the computation workload inside each function was firstly obtained running the sequential application on a system including one Microblaze processor implemented on FPGA. Using a dedicated performance counter it was possible to measure the execution time associated to the main steps of the elaboration.

In Table 7.1 the results of such profiling have been presented. The communication latency is an estimated value obtained by means of a simple analytic model derived from a preliminary training set of experiments. It provides an estimation of the latency associated with the reading and the writing operations that should be performed if a node has to read inputs and write outputs through FIFOs, according to the size of the tokens that have to be exchanged. The number of iterations reported in Table 7.1 indicates how many times each function has to be executed to process one spike. The first three nodes iterate over each sample. Each iteration can prospectively determine the need for a spike analysis. This part of the processing, thus, is obviously not limiting the overall throughput.

Table 7.1: Execution time of the main processing steps.

Elaboration step	Exec. time (cycles)	Comm. latency	iterations
Wavelet denoising	1981	25	1
NEO extraction	210	25	1
Spike detection	253	450	1
Spike standardization – 1	55584	1800	1
Spike standardization – 2	62434	26100	1
SpikeVStemplate xcorr	22001	25700	N_t
Maximum search	45	$15 \times N_t$	1

The profiling clearly shows that the most computationally intensive processing phase in the algorithm is the spike sorting. Considering it as a single node would result in an unbalanced partitioning reducing the achievable speed-up. Thus it has been considered the sub-steps involved in the spike sorting as independent functions. The spike standardization is performed once for every overlap between a template and the overlapping part of the detected spike, whereas the spike-template cross-correlation measurement has to be repeated also for each reference template, thus exposing further parallelism.

7.6 Partitioning and mapping description

The chosen partitioning involves that the standardization process is decomposed into two different steps considered as independent nodes and the cross-correlation measurement into N_t nodes, where N_t is the templates number previously created at off-line. In this way, the parallel application graph assumes the form represented in Fig. 7.3. The first phase of the spike standardization computes the required statistic moments, namely the mean and the variance of the segment of the enlarged spike buffer used for that cross-correlation step. The second phase derives the standard deviation and performs the actual standardization of that part of the enlarged buffer, preparing the segment for the next processing step.

Looking at the entry Table 7.1 related to the second phase of the spike standardization, it is easy to notice that such node has to communicate a significant amount of data with the downstream nodes. This can prospectively be a criticality of the proposed partitioning. A possible countermeasure could be the integration of the second standardization phase in each downstream node, which would reduce significantly the communication requirements (the communication execution time overhead would be reduced from 25700 to around 1800 as in the upstream communication link). However, the spike standardization is not dependent on the reference template, thus, using this solution, each *SpikeVsTempalte xcorr* downstream node would repeat the same computation on the same input data, wasting FLOPs (Floating-Point Operations) in the assigned processing elements.

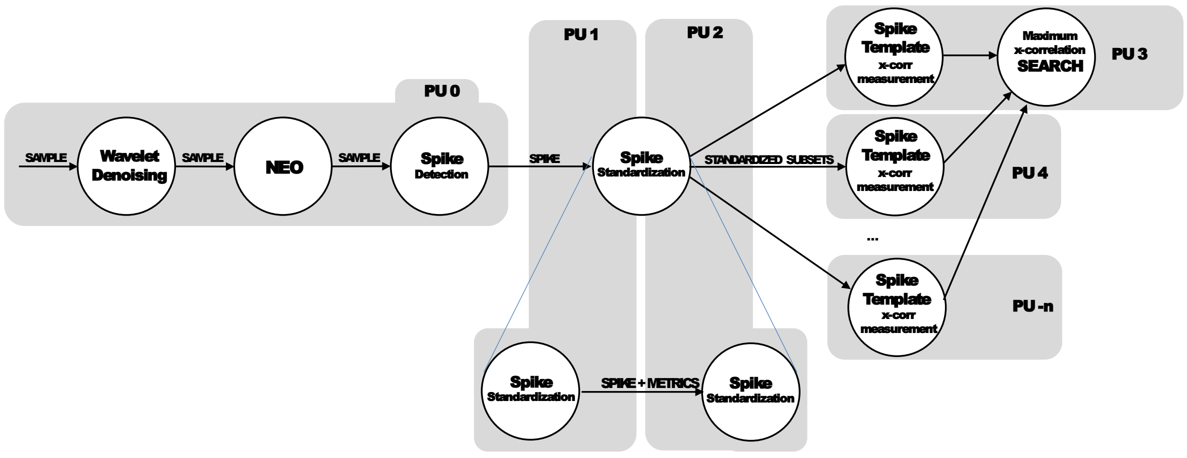


Figure 7.3: Application graph resulting from the application partitioning. The gray boxes indicate the task-to-processor mapping.

The task-to-processor mapping has to provide a good balancing of the workload distribution over the different processing elements. The *Spike standardization - 2* node has the heaviest workload. Selecting it as the limiting node for the overall throughput sets an execution time limit for the other nodes. To respect this limit, it has been selected the mapping represented by the gray boxes in Fig. 7.3. The first three nodes are merged on one single node and mapped on the processing unit *PU0*. On *PU1* and *PU2* it has been mapped respectively *Spike standardization - 1* and *Spike standardization - 2*. The rest of the processors are used as *Normalized cross-correlators*, each one performing an instance of the *SpikeVsTemplate xcorr*, over the incoming spike and one of the reference templates.

According to the latency associated with the *SpikeVsTempalte xcorr* task, given the proposed mapping, it is possible to observe that two different iterations of the node can be performed without impacting on the overall throughput of the network. Thus it is possible to assume that, if needed, each *normalized cross-correlator* can measure the cross-correlation of the incoming spike with two different reference templates. Finally, the *spike sorting* phase that searches for the maximum cross-correlation value is assigned to one of the *normalized cross-correlators*.

7.7 Hardware architecture

Given the previously described partitioning of the application, to implement the computing platform on the FPGA, the Embedded Development Kit tool-suite by Xilinx has been exploited. As shown in Fig. 7.4, 8 tiles have been instantiated in the multi-processor system. Each tile is composed of a Microblaze processor connected by means of a Local Memory Bus to a double-port memory (two ports implement the instruction and data sides). Each tile is connected to the neighbours, according to the communication needs posed by the task graph represented in Fig. 7.3, by means of Fast Simplex Link (FSL) FIFO-based structures.

The FIFO links are 32-bit wide. The transmission of tokens is implemented using a dedicated API that, according to the size specified as parameter, wraps a variable number of the dedicated assembly primitives available in the Microblaze instruction set. In the shaded area of Fig. 7.4, it has been highlighted the part of the system that has been implemented on FPGA only for prototyping purposes. It includes a PLB (Peripheral Local Bus), providing to the PEs access to a set of shared peripherals (a serial I/O to enable communication with a host workstation, a performance counter exploited during the profiling, a DDR2 interface that is exploited by the first node to emulate the access to the analog back-end in order to acquire the input signal samples coming from the recording phase). Hence, the hardware parts in the shaded area are only useful to demonstrate the functionality of the system but will not be needed in a prospective industrial implementation of the computing platform.

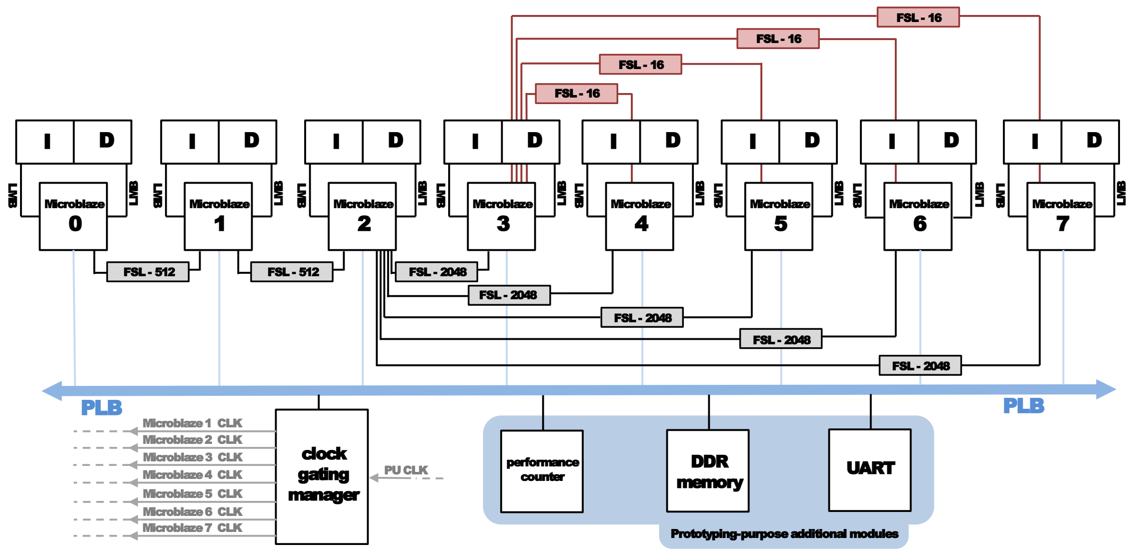


Figure 7.4: Custom MPSoC architecture implemented on the FPGA device.

7.8 Integrated power consumption control

The overall approach is based on the already mentioned structure of the input neural signal. The idea is to provide enough computing power to process in real-time bursts of very frequent spikes, making the application robust and secure in the very worst-case condition, but being capable of reducing the power consumption when the hardware elements are not used, in those time slots where neural activity is below the background activity. To this aim, a clock-gating manager has been implemented that can selectively switch off the clock signals in input to the processing elements and the communication infrastructures that have been mapped into the last stages of the processing pipeline. They usually determine the greatest contribution in terms of dynamic power consumption due to the fact that the relative internal registers are updated at each clock period even if their input signals do not change when they are not involved in the current processing.

In digital systems, power consumption is in fact composed of two contributions: dynamic and static. The former is mainly due to the charging and recharging of parasitic capacitances when logic transitions occur (i.e. switching-activity during execution) and it is influenced by the characteristics of the reference application in terms of probability with which the various internal wires of the system change their logic state. The latter is dissipated while no circuit activity is present in the system and is due to leakage currents and it depends on the used technology library. As said before, the aim is to strongly reduce the former by using clock-gating techniques according to considerations performed on the signal of interest.

The clock-gating manager can be controlled by calling a set of related APIs in the application software. It has been integrated in a peripheral module shared among the various processing units which can access it through the same PLB. According to the current state of a Finite State Machine (FSM), the clock signal in input to each core is enabled or disabled assigning the correct logic value to a 1-bit wire in input to the dedicated *BUFGCE* block available on Xilinx FPGAs whereas in case of ASIC implementations, it can be performed easily by exploiting AND gates having in input the enable and the clock signals.

In the presented solution, the clock signals of the processors in charge of analysing the spikes are initially disabled at the application start-up. The detection stage, when a spike is identified in the output of the Wavelet Denoising stage, enables the clock applied to the downstream pipeline stage. Then the activation process continues similarly: each processing stage activates the following right before writing the data to be processed in the related connecting FIFO. Every process than takes care of disabling its own clock when the received token has been processed. In order to avoid such auto-disabling to overwrite previous activations received by the upstream node during the processing phase, the clock-gating manager hardware has been designed to take care of tracking the history of activations and disactivations. Thus, it ensures that all the tokens are processed.

The implemented clocking manager is parametric. The number of clocks to control can be configured at design time. Thus the designer can choose to control each "Normalized cross-correlator" independently. This feature is useful since the template definition can be repeated periodically to update and refine the shapes and the number of the templates that have to be compared with the detected spike. Thus, if the number of spikes to check is reduced, the application can activate only the needed number of Normalized cross-correlators and can keep the power consumption as low as possible. Fig. 7.5 shows how it is possible to reduce the power consumption of the system implemented on a Xilinx FPGA Virtex-5 LX50T by means of the proposed clock gating techniques. The plot represents the power levels that can be selected, when different numbers of Normalized cross-correlators are enabled. The trend of the instantaneous consumption is obtained exploiting the *Digilent Adept* tool which allows to perform the software real-time power monitor on all the supply rails after the proper configuration of the designed MPSoC system into the target FPGA device, buffering the relative samples for a defined time window at the frequency of $16Hz$.

While only clock gating can be implemented using FPGAs as target technology, the proposed system can serve as a proof of concept with respect to the usefulness of the considered approach for the perspective of a future ASIC implementation, where a more aggressive switching policies, involving gating of power islands can be adopted in order to allow portability of the resultant prosthetic solution with a long battery life. In fact, the achieved absolute values of the instantaneous power consumption of the system mapped into the FPGA shown in Fig.7.5 do not have significant importance whereas the relative percentage variation after the shut-down process of one or more processing units strongly highlights the utility of this approach. The KPN nodes that can be switched off are stateless, thus no information has to be stored between two successive firings. In this way, switching off the whole processor power supply is totally feasible.

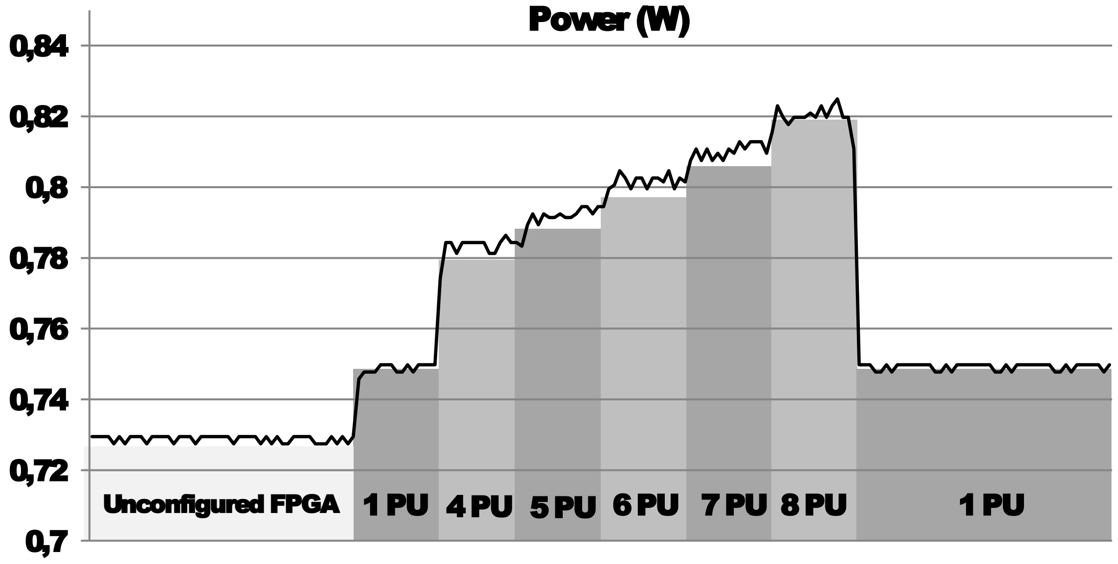


Figure 7.5: Power consumption of the system measured on the FPGA. The background bars indicate the number of non clock-gated CPUs in each time interval.

7.9 Experimental results

7.9.1 Timing Constraints Evaluation

To verify the actual performance of the implemented prototype and the correct functionality even in worst-case conditions, evaluating that the balance of the tasks mapped on each processing unit of the system matches with the expected results after the profiling, a synthetic test signal has been used. It has been created from a publicly available dataset of synthetic neural signal [72] as a continuous sequence of the same real physiological spike. This spike has been selected with the minimum allowable number of samples according to physiological considerations in order to stimulate the worst case condition for the decoding algorithm throughput requirements and to verify the satisfaction of the real-time constraints in the FPGA MPSoC system. As said before, for each instance of the considered spike in the input signal, the hardware implementation must be able to determine the output class at real-time without any loss of information.

The timing diagram in Fig. 7.6 demonstrates that the proposed architecture is able to meet the constraints imposed by the application. To do this, the *Grasp* [31] software has been exploited which represents a tool for tracing, visualizing and measuring the behaviour of hierarchical multi-processor real-time systems. The resulting throughput, tested during the on-hardware execution, is obtained by means of a performance counter accessible in the architecture by the same PLB which is read by each processor every time the relative *compute* part respectively starts and finishes. The time interval between the end of the *compute* part mapped on a defined processor and the starting of that of the next processor in the pipeline depends on the amount of data which must be transmitted among them. Due to the fact that the spikes in the test signal are too close, in the Fig. 7.6 it is not possible to see the same interval for the first processor of the pipeline which is in charge of detecting the various pulses and send them to the next processing stage.

The throughput is, as expected, limited by that of the pipeline node performing the phase 2 of the spike standardization, which is the most critical in terms of computational complexity according to the used process mapping. The system is able to elaborate around one spike every 89 kcycles and, to meet the worst-case constraint of 524 spikes/second defined by the need of processing pulses with at least 23 samples at the input sampling frequency of 12kHz, the clock frequency has to be set at least to 47 MHz. It allows to further reduce the dynamic power consumption of the proposed solution in comparison to the previous DSP implementation.

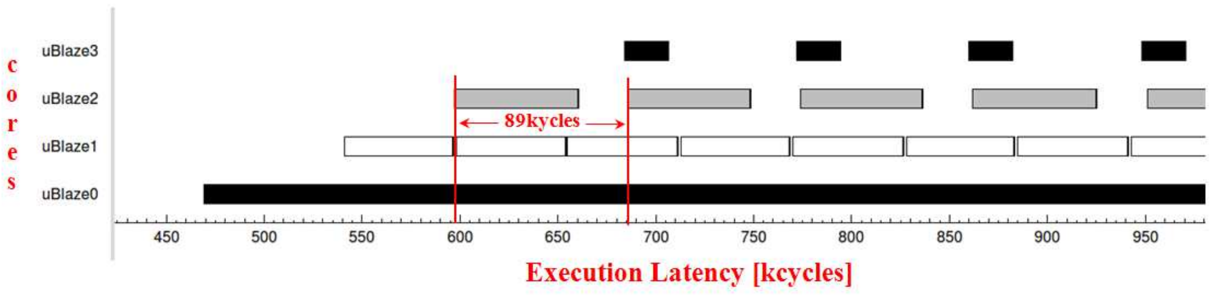


Figure 7.6: Execution trace of the multi-core system.

7.9.2 Power consumption reduction

Fig. 7.5 shows the power consumption levels that can be set for the architecture. The eventual effectiveness of the power reduction technique is obviously dependent on the rate at which the spikes are detected in real neural signals. To evaluate it, the target application has been executed to analyse a test set of physiologically plausible signals [72].

To this aim, the *Easy1* dataset of synthetic neural signals constructed starting from a database of real physiological spikes extracted from neocortex and basal ganglia at CNS level has been used considering that each signal in the dataset has a duration time of 60 sec. It has been detected, by means of a dedicated counter instantiated in the clock-gating manager for prototyping purposes, the percentage of cycles during which the clock to the sorters was actually enabled. The obtained percentages are reported in Table 7.2.

Table 7.2: Percentage of time with enabled Normalized cross-correlators

Signal	Cycles % with disabled clock gating
Easy1-noise005	11.05%
Easy1-noise01	11.73%
Easy1-noise015	11.44%
Easy1-noise02	10.69%
Easy1-noise025	9.71%
Easy1-noise03	8.97%
Easy1-noise035	9.78%
Easy1-noise04	8.24%
unpublished PNS recordings	10.09%

Thanks to sparse nature of the neural signals, it can be noticed that the Normalized cross-correlators are inactive for most of the whole processing time. The possibility of switching off their clock at runtime results in a significant impact of the proposed approach. Such impact is expected to be even greater in an ASIC implementation where power gating can also be exploited to cut off also the static part of the power consumption. It allows portability for the resulting solution and a longer battery life.

7.9.3 Synthesis Results

The previous results demonstrate the applicability of the proposed approach to create an FPGA hardware implementation able to perform real-time neural signal decoding for neuroprosthetic applications. Such results have been achieved starting from the hypothesis that there is enough availability of slices and memory resources (blocks of RAMs) to allow the instantiation of the various processors into the considered board. In particular, due to the computational complexity of the cross-correlation measurements which must be done in parallel, it is necessary to have a number of processors equal to N_t+3, where N_t represents the templates number, together with the logic to handle the communication among them according to the implemented scheme shown in Fig. 7.4 and the dedicated local memory to store the relative text code and the data structures to perform the correspondent task.

As it has been analysed before, the processing using a sample-by-sample approach in comparison to the block-based one has allowed to minimize the requirements of each processor in terms of memory resources thanks to the fact that they don't need large data structures and large tokens which must be transmitted among them. In this subsection, the hardware resources requirements which the MPSoC solution needs are properly evaluated by FPGA synthesis exploiting the dedicated Xilinx Synthesis Tool (XST) and different FPGAs belonging to the same Virtex-5 architecture as target devices at the varying of the number of templates N_t.

Table 7.3 shows the number of Slice Registers, Slice LUTs, DSP48E devices and Block RAMs which are necessary for the various processors instantiated in the considered system, at the varying of the templates number N_t, considering the relative requirements according to the task which each one must execute, analysing the synthesis reports generated by XST. They allow to demonstrate how in a Xilinx FPGA Virtex-5 LX50T it is possible to map up to $N_t = 5$ processors performing in parallel way the cross-correlation calculation between the

Table 7.3: FPGA Synthesis

Resources	**uBlaze0**			**uBlaze1**			**uBlaze2**			**uBlaze3/4**		
	Core	**Memory**	**FSL link**	**Core**	**Memory**	**FSL link**	**Core**	**Memory**	**FSL link**	**Core**	**Memory**	**FSL link**
Slice Registers	2807	6	29	2574	6	29	2582	6	$35\times N_t$	1842	6	0/7
Slice LUTs	2885	12	469	2572	12	469	2789	12	$1819\times N_t$	1592	12	0/44
DSP48Es	6	0	0	6	0	0	6	0	0	4	0	0
Block RAMs	0	16	0	0	4	0	0	8	0	0	4	0

current spike and a defined template, whereas in a Virtex-5 LX110T device the maximum value for N_t can be fixed up to 25. This constraint for the two considered devices are imposed by the needs in terms of memory requirements for the instantiated processors which determine a percentage usage of 100% of Block RAMs.

Typically, the number of classes per channel is limited by the selectivity of the electrode up to 3-4 in the very best case, more otherwise, so the number of Normalized cross-correlation processors which must be used to satisfy the real-time constraints of the application can be very low. However, with this approach, the multi-core systems can contain the maximum number of functional units working in parallel, determining the use of a worst-case architecture, enabling or not some of them based on the current considered N_t adopting the clock gating approach.

Summarizing, the proposed implementation enables to face the challenge posed by the data-dependence in the target application. Enough processing power is instantiated to respect real-time constraints in the worst case operating condition. The needed computation capabilities can be achieved setting a reasonable operating clock frequency, lower than $50MHz$. To counterbalance the overprovisioning of the system deriving from the worst-case assumption, an integrated power management relying on software-controlled clock-gating has been implemented and tested. Such feature allows saving a significant energy consumption related with dynamic power in the FPGA-based prototype, and demonstrates an even greater prospective usefulness of similarly controlling more aggressive techniques like power gating in an ASIC implementation of the system. Such a study could lead to the realization of an integrated decoding architecture potentially able to be hosted in the stump of the amputee using batteries as power source, in this way addressing the main issue motivating the choice of this specific application.

Chapter 8

Conclusions

In this thesis, the design issues concerning the development of a neuroprosthetic device able to restore the functionalities lost by an upper limb amputee in the common daily life have been addressed, trying to give a practical solution for each of the various phases involved in the reference closed-loop system without the use of complex laboratory instrumentations. The efforts of the research have been especially focused in the hardware implementation of the electronic devices needed in the direct path for the recording and processing of peripheral neural signals in order to decode the movements intentions, trying to create a communication link between the brain and the cybernetic hand, and for the stimulation patterns generation in the opposite one to give sensory information about the surrounding world to the patient subjected to this particular disease. The research lack that this work wants to cover is the necessity of satisfying the tight constraints imposed by the application under consideration on embedded systems with limited resources to finally obtain a solution for the optimal ENG-based approach which can have a great dissemination from the academic and the commercial point of view.

In particular, a sigma-delta architecture for the recording of significant neural signals at PNS level has been conceived, designed and simulated. It is composed by an analog front-end unit (bandpass pre-filtering and sigma-delta modulator) modelled at behavioural (Matlab Simulink) and transistor level and by a digital back-end hosted on a Xilinx FPGA Virtex-5 LX330 and tested together with the analog part by hardware/software cosimulations exploiting the Xilinx System Generator tool. The analog module amplifies and filters the weak neural signal as close as possible to the recording site and converts it in a digital bit stream by means of a third order single loop sigma-delta modulator. The digital unit provides sigma-delta decimation and downsampling at the Nyquist rate as well as configuration and control of the analog part. In this way, it has been demonstrated that the converter is able to totally remove the unwanted EMG interferences thanks to the high integration capacity of the digital domain, obtaining a good output resolution allowing to detect signals in the order of magnitude of tens of microvolt.

The same architecture has been then used to implement a novel 8-channel bidirectional interface, aimed at the recording of the peripheral neural signals coming from the electrodes and the generation of bi-phasic stimulation pulses. Even in this case, the resulting system has been designed by two separated parts: the analog front-end mapped on a custom designed Integrated Circuit which exhibits a maximum power consumption of $27.2mW$ and a total area occupation of $16.8mm^2$, implementing the band-pass pre-filtering and the sigma-delta modulation for the recording phase and eight D/A converters for stimulation, which can be implanted near the electrodes avoiding to add huge noise due to long wires connections; the digital back-end hosted into a Xilinx FPGA Spartan-3E prototyping board performing the multi-channel sigma-delta decimation and controlling the configuration of the analog chip. The device has been successfully tested by means of in-vivo experiments with rats, in which an eight-channels TIME electrode has been chronically implanted.

Regarding to the decoding phase, it is necessary that the dedicated hardware implementation hosted in the stump of the amputee is able to ensure the correctness of the required functionality even in case of critical conditions, performing the complex processing in real-time on portable low-power devices with limited resources operating at a frequency compliant with respect to the constraints imposed by the target application. To this aim, a complete study of the optimization of a state-of-the-art algorithm for PNS signals decoding has been deeply analysed in order to obtain the best performance on an embedded platform such as an off-the-shelf DSP. Several tests have been performed both on synthetic neural datasets and on real afferent signals recorded in-vivo from rodents allowing to achieve an accuracy up to 96% in classification. The main implementation issues regarding the porting of the whole processing algorithm on the target floating-point DSP platform, allowing the fulfilment of real-time constraints, have been discussed. In a single-channel implementation, the algorithm is in fact able to process up to 400 spikes per second when the unsupervised templates creation procedure is running, and up to 1600 in an use case where 10 templates are required to obtain the pattern features. Such numbers have evaluated and demonstrated the possible extension to a multi-channel scenario involving closed-loop real-time experiments including the complex phase of classifier training, included in the same embedded framework so that also the training phase could be carried out without any external tool.

However, when the real-time requirements are joined to the fulfilment of area and power minimization for portable/implantable solutions only custom VLSI implementations can be adopted. In this case, every part of the algorithm should be carefully retuned. For this reason, this process has been initially performed for the first Wavelet Denoising stage for which the choice of the threshold estimation technique, more than having an influence on the quality of the denoising, has significant reverberations on the feasibility and efficiency of a custom VLSI architecture. The comparison between a sample standard deviation and the widespread median absolute deviation has revealed similar functional performance with dramatically better characteristic of the former in terms of hardware implementation, regardless the MAD is implemented as a combinatorial trellis or in a more efficient folded version.

Finally, two different hardware implementations of the reference decoding algorithm have been presented, highlighting pros and cons of each one of them. Initially, a novel approach based on high-level dataflow description and automatic hardware generation has been presented and evaluated on the on-line template-matching spike sorting algorithm. Results in the best case have revealed a 71% of area saving compared to more traditional solutions thanks to hardware resources sharing to perform different kernels with higher computational complexity, without any accuracy penalty. Better latency performance have been achieved still minimizing the number of adopted resources even if these can also be improved by tuning the implemented parallelism in the light of a defined number of channels and real-time constraints. It can be done by using more than one reconfigurable platform in order that they can be exploited to perform the same or different kernels at the same time in a parallel way, due to the fact that each one can execute the relative processing only in a sequential way.

For this reason, a second FPGA solution have been proposed based on the use of a Multi-Processor System-on-Chip (MPSoC) embedded architecture. It has been verified that this prototype is capable of respecting the real-time constraints posed by the application when clocked at less than 50 MHz, in comparison to 300 MHz of the previous DSP implementation. Considering that the application workload is extremely data dependent and unpredictable due to the sparsity of the neural signals, the architecture has been dimensioned taking into account critical worst-case operating conditions in order to always ensure the correct functionality. To compensate the resulting over-provisioning of the system architecture, a software-controllable power management based on the use of clock gating techniques has been integrated in order to minimize the dynamic power consumption of the resulting solution.

At first sight, it might seem that the objective of the development of a portable, low-power and real-time hardware implementation of the neuro-controlled prosthetic device has been correctly reached in all the design aspects but, however, there are still a lot of open problems which must be addressed. Probably, the most important one of this research topic is due to the difficulty of performing experimental tests on animals or humans in order to verify and optimize the performance of the resulting solutions because every time it is necessary to obtain a specific consensus by the ethical committee. Moreover, it is fundamental to evaluate how the results can change or degrade when the proposed implementations for each of the various phases involved in the closed-loop system are assembled together. This is the next step which must developed at short term. In particular, it must be evaluated how its behaviour can evolve based on the current characteristics of the acquired signal which can be very critical in case of a chronic implantation, for example due to the growth of cellular tissues around the position in which the electrodes are placed in the peripheral nerves, increasing the relative input impedance. For these reasons, this research topic can be one of the most attractive on which much progress still to be made before to achieve a great dissemination but this contribution can be considered an important milestone among the several works presented in the literature during the years.

Bibliography

[1] ISO/IEC 23001-4 (2009).MPEG systems tech.—Part 4: Codec configuration representation. [cited at p. 86]

[2] Otto Bock Healthcare, Minneapolis, MN. [cited at p. 2]

[3] Touch EMAS Ltd, Edinburgh, U.K. [cited at p. 2]

[4] K. Azadet and C.J. Nicole. Low-power equalizer architectures for high speed modems. *Communications Magazine, IEEE*, 36(10):118–126, 1998. [cited at p. 98]

[5] M. Bahoura, M. Hassani, and M. Hubin. DSP implementation of wavelet transform for real time ECG wave forms detection and heart rate analysis. *Computer Methods and Programs in Biomedicine Volume*, 52(1):35–44, January 1997. [cited at p. 72]

[6] Mohammed Bahoura and Hassan Ezzaidi. FPGA-implementation of discrete wavelet transform with application to signal denoising. *Circuits, Systems, and Signal Processing*, 31(3):987–1015, 2012. [cited at p. 73, 76]

[7] Mirza Mansoor Baig, Hamid Gholamhosseini, and Martin J. Connolly. A comprehensive survey of wearable and wireless ECG monitoring systems for older adults. *Medical & Biological Engineering & Computing*, 51(5):485–495, 2013. [cited at p. 71]

[8] K. Balasubramanian and I. Obeid. Reconfigurable embedded system architecture for next-generation neural signal processing. In *Engineering in Medicine and Biology Society EMBC, 2010 Annual International Conference of the IEEE*, pages 1691–1694, 2010. [cited at p. 85, 102]

[9] Marcelo Blatt, Shai Wiseman, and Eytan Domany. Superparamagnetic clustering of data. *Physical review letters*, 76(18):3251–3254, 1996. [cited at p. 61]

[10] B. Boser, I. Guyon, and V. Vapnik. An training algorithm for optimal margin classifiers. In *Proc. Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. [cited at p. 55]

[11] A.J. Casson, D. Yates, S. Smith, J.S. Duncan, and E. Rodriguez-Villegas. Wearable electroencephalography. *Engineering in Medicine and Biology Magazine, IEEE*, 29(3):44–56, 2010. [cited at p. 71]

[12] M.S. Chae, Zhi Yang, M.R. Yuce, Linh Hoang, and W. Liu. A 128-channel 6 mw wireless neural recording ic with spike feature extraction and uwb transmitter. *IEEE Transaction on Neural Systems and Rehabilitation Engineering*, 17(4):312–321, 2009. [cited at p. 21]

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. [cited at p. 58]

[14] Dan Chen, Lizhe Wang, Gaoxiang Ouyang, and Xiaoli Li. Massively parallel neural signal processing on a many-core platform. *Computing in Science & Engineering*, 13(6):42–51, 2011. [cited at p. 103]

[15] Tung-Chien Chen, Wentai Liu, and Liang-Gee Chen. 128-channel spike sorting processor with a parallel-folding structure in 90nm process. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 1253–1256, May 2009. [cited at p. 102]

[16] Luca Citi, Jacopo Carpaneto, Ken Yoshida, Klaus-Peter Hoffmann, Klaus Peter Koch, Paolo Dario, and Silvestro Micera. On the use of wavelet denoising and spike sorting techniques to process electroneurographic signals recorded using intraneural electrodes. *Journal of Neuroscience Methods*, 172:294–302, 2008. [cited at p. 2, 45, 46, 50, 55, 56, 57, 61, 64, 69, 72, 73, 101]

[17] G. S. Dhillon and K. W. Horch. Direct neural sensory feedback and control of a prosthetic arm. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, pages 468–472, dec 2005. [cited at p. 21]

[18] A Diedrich, W Charoensuk, R.J. Brychta, A.C. Ertl, and R. Shiavi. Analysis of raw microneurographic recordings based on wavelet de-noising technique and classification algorithm: wavelet analysis in microneurography. *IEEE Trans Biomed Eng*, 50(1):41–50, 2003. [cited at p. 72, 101]

[19] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. [cited at p. 51, 52]

[20] Stefano Fusi, Mario Annunziato, Davide Badoni, Andrea Salamon, and Daniel J Amit. Spike-driven synaptic plasticity: theory, simulation, vlsi implementation. *Neural Computation*, 12(10):2227–2258, 2000. [cited at p. 22]

[21] H. Gao, R.M. Walker, P. Nuyujukian, K.A.A. Makinwa, K.V. Shenoy, B. Murmann, and T.H. Meng. Hermese: A 96-channel full data rate direct neural interface in 0.13 μm cmos. *IEEE Journal of Solid-State Circuits*, 47(4):1043–1055, 2012. [cited at p. 21]

[22] Marc Geilen and Twan Basten. Requirements on the execution of kahn process networks. In Pierpaolo Degano, editor, *Programming Languages and Systems*, volume 2618 of *Lecture Notes in Computer Science*, pages 319–334. Springer Berlin Heidelberg, 2003. [cited at p. 108]

[23] S. Gibson, J.W. Judy, and D. Markovic. Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(5):469–478, October 2010. [cited at p. 52, 103]

[24] Sarah Gibson, Jack W. Judy, and Dejan Markovic. An fpga-based platform for accelerated offline spike sorting. *Journal of Neuroscience Methods*, 215(1):1 – 11, 2013. [cited at p. 85, 103]

[25] Graphiti Editor, . [cited at p. 88]

[26] R. R. Harrison. A versatile integrated circuit for the acquisition of biopotentials. *Custom Integrated Circuits Conference*, pages 115–122, 2007. [cited at p. 7]

[27] R. R. Harrison and C. Charles. A low-power low-noise cmos amplifier for neural recording applications. *IEEE Journal of Solid-State Circuits*, 38:958–965, 2003. [cited at p. 7, 21]

[28] R. R. Harrison, R. J. Kier, C. A. Chestek, V. Gilja, P. Nuyujukian, S. Ryu, B. Greger, F. Solzbacher, and K. V. Shenoy. Wireless neural recording with single low-power integrated circuit. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pages 322–329, 2009. [cited at p. 21]

[29] Jiping He, Chaolin Ma, and R. Herman. Engineering neural interfaces for rehabilitation of lower limb function in spinal cord injured. *Proceedings of the IEEE*, 96(7):1152–1166, 2008. [cited at p. 21]

[30] E Hogenauer. An economical class of digital filters for decimation and interpolation. *IEEE Transactions on Acoustics Speech and Signal Processing*, 29(2):155–162, 1981. [cited at p. 14, 15]

[31] Mike Holenderski, Reinder J Bril, and Johan J Lukkien. Grasp: Visualizing the behavior of hierarchical multiprocessor real-time systems. *Journal of Systems Architecture*, 59(6):307–314, 2013. [cited at p. 114]

[32] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, March 2002. [cited at p. 56]

[33] IEEE Task P754. *ANSI/IEEE 754-1985, Standard for Binary Floating-Point Arithmetic.* August 1985. [cited at p. 89]

[34] T. Jochum, T. Denison, and P. Wolf. Integrated circuit amplifiers for multi-electrode intracortical recording. *J. Neural Eng.*, 6:1–26, 2009. [cited at p. 7]

[35] G. Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information processing*, pages 471–475, Stockholm, Sweden, Aug 1974. North Holland, Amsterdam. [cited at p. 107]

[36] J.F. Kaiser. On a simple algorithm to calculate the 'energy' of a signal. In *Proc. International Conference on Acoustics, Speech, and Signal Processing, ICASSP-90*, volume 1, pages 381–384, April 1990. [cited at p. 52, 105]

[37] V. Karkare, S. Gibson, and D. Markovic. A 130- μ w, 64-channel neural spike-sorting dsp chip. *Solid-State Circuits, IEEE Journal of*, 46(5):1214–1222, May 2011. [cited at p. 103]

[38] W. Kester. Mixed signal and dsp design techniques. *Analog Devices, Inc.*, 2000. [cited at p. 15]

[39] Todd A. Kuiken, Guanglin Li, Blair A. Lock, Robert D. Lipschutz, Laura A. Miller, Kathy A. Stubblefield, and Kevin B. Englehart. Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms. *JAMA*, 301(6):619–628, 2009. [cited at p. 3]

[40] Todd A. Kuiken, Laura A. Miller, Robert D. Lipschutz, Blair A. Lock, Kathy Stubblefield, Paul D. Marasco, Ping Zhou, and Gregory A. Dumanian. Targeted reinnervation for enhanced prosthetic arm function in a woman with a proximal amputation: a case study. *Lancet*, 369:371–380, February 2007. [cited at p. 3]

[41] Koichi Kuzume, Koichi Niijima, and Shigeru Takano. FPGA-based lifting wavelet processor for real-time signal detection. *Signal Processing*, 84(10):1931–1940, 2004. [cited at p. 72]

[42] Natalia Lago, Dolores Ceballos, Francisco J Rodrı?guez, Thomas Stieglitz, and Xavier Navarro. Long term assessment of axonal regeneration through polyimide regenerative electrodes to interface the peripheral nerve. *Biomaterials*, 26(14):2021–2031, 2005. [cited at p. 4]

[43] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. Incremental support vector learning: Analysis, implementation and applications. *The Journal of Machine Learning Research*, 7:1909–1936, 2006. [cited at p. 56]

[44] J. Lee, H. G. Rhew, D. R. Kipke, and M. P. Flynn. A 64 channel programmable closed-loop neurostimulator with 8 channel neural amplifier and logarithmic adc. *IEEE Journal of Solid-State Circuits*, 45:1935–1945, 2010. [cited at p. 21]

[45] S. Y. Lee and S. C. Lee. An implantable wireless bidirectional communication microstimulator for neuromuscolar stimulation. *IEEE Trans. Circuit System*, 52:2526–2538, 2005. [cited at p. 8]

[46] M. S. Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network*, 9(4):R53–R78, November 1998. [cited at p. 2, 102]

[47] K. Limnuson, D. J. Tyler, and P. Mohseni. Integrated electronics for peripheral nerve recording and signal processing. *31st Annual International Conference of the IEEE EMBS*, pages 1639–1642, 2009. [cited at p. 8]

[48] X. Liu, A. Demosthenous, A. Vanhoestenberghe, Dai Jiang, and N. Donaldson. Active books: The design of an implantable stimulator that minimizes cable count using integrated circuits very close to electrodes. *IEEE Transaction on Biomedical Circuits and Systems*, 6(3):216–227, 2012. [cited at p. 21]

[49] GE Loeb and RA Peck. Cuff electrodes for chronic stimulation and recording of peripheral nerve activity. *Journal of neuroscience methods*, 64(1):95–103, 1996. [cited at p. 4]

[50] D. Loi, C. Carboni, G. Angius, G.N. Angotzi, M. Barbaro, L. Raffo, S. Raspopovic, and X. Navarro. Peripheral neural activity recording and stimulation system. *Biomedical Circuits and Systems, IEEE Transactions on*, 5(4):368–379, 2011. [cited at p. 21]

[51] Mohamed I. Mahmoud, Moawad I. M. Dessouky, Salah Deyab, and Fatma H. Elfouly. Signal denoising by wavelet packet transform on FPGA technology. *Special Issue of Ubiquitous Computing and Communication Journal of Bioinformatics and Image*, 2008. [cited at p. 73]

[52] P. Malcovati, S. Brigati, F. Francesconi, F. Maloberti, P. Cusinato, and A. Baschirotto. Behavioral modeling of switched-capacitor sigma delta modulators. *IEEE Trans. on Circuits and Systems-I*, 5:352–364, 2003. [cited at p. 12]

[53] J. Martinez, R. Cumplido, and C. Feregrino. An FPGA-based parallel sorting architecture for the Burrows Wheeler transform. In *International Conference on Reconfigurable Computing and FPGAs, ReConFig 2005*, 2005. [cited at p. 77]

[54] C.A. Medina, A. Alcaim, and J.A. Apolinario Jr. Wavelet denoising of speech using neural networks for threshold selection. *Electronics Letters*, 39(25):1869–1871, 2003. [cited at p. 72]

[55] S. Meijer, H. Nikolov, and T. Stefanov. Throughput modeling to evaluate process merging transformations in polyhedral process networks. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 747–752, 2010. [cited at p. 108]

[56] S. Micera, J. Carpaneto, and S. Raspopovic. Control of hand prostheses using peripheral information. *IEEE reviews in biomedical engineering*, 3:48–68, 2010. [cited at p. 103]

[57] S. Micera, L. Citi, J. Rigosa, J. Carpaneto, S. Raspopovic, G. Di Pino, L. Rossini, K. Yoshida, L. Denaro, P. Dario, and P. M. Rossini. Decoding information from neural signals recorded using intraneural electrodes: Toward the development of a neurocontrolled hand prosthesis. *Proceedings of the IEEE*, 98(3):407–417, 2010. [cited at p. 21]

[58] S Micera, PN Sergi, J Carpaneto, L Citi, S Bossi, K-P Koch, KP Hoffmann, A Menciassi, Ken Yoshida, and P Dario. Experiments on the development and use of a new generation of intraneural electrodes to control robotic devices. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 2940–2943. IEEE, 2006. [cited at p. 5]

[59] Silvestro Micera, Xavier Navarro, Jacopo Carpaneto, Luca Citi, Oliver Tonet, Paolo Maria Rossini, Maria Chiara Carrozza, Klaus Peter Hoffmann, Meritxell Vivo, Ken Yoshida, et al. On the use of longitudinal intrafascicular peripheral interfaces for the control of cybernetic hand prostheses in amputees. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 16(5):453–472, 2008. [cited at p. 10, 46]

[60] Silvestro Micera, Paolo M Rossini, Jacopo Rigosa, Luca Citi, Jacopo Carpaneto, Stanisa Raspopovic, Mario Tombini, Christian Cipriani, Giovanni Assenza, Maria C Carrozza, Klaus-Peter Hoffmann, Ken Yoshida, Xavier Navarro, and Paolo Dario. Decoding of grasping information from neural signals recorded using peripheral intrafascicular interfaces. *Journal of NeuroEngineering and Rehabilitation*, 8(53):1038–1051, 2011. [cited at p. 2]

[61] M. Montani, L. De Marchi, A. Marcianesi, and N. Speciale. Comparison of a programmable DSP and FPGA implementation for a wavelet-based denoising algorithm. In *Proc. IEEE 46th Midwest Symposium on Circuits and Systems*, volume 2, pages 602–605, 2003. [cited at p. 71, 103]

[62] Gernot R. Muller-Putz, Reinhold Scherer, Gert Pfurtscheller, and Rudiger Rupp. EEG-based neuroprosthesis control: A step towards clinical practice. *Neuroscience Letters*, 382(1?2):169 – 174, 2005. [cited at p. 2]

[63] X. Navarro, T.B. Krueger, N. Lago, S. Micera, T. Stieglitz, and P. Dario. A critical review of interfaces with the peripheral nervous system for the control of neuroprostheses and hybrid bionic systems. *J Peripher Nerv Syst*, 10(3):229–258, 2005. [cited at p. 21, 106]

[64] J. Nezan, N. Siret, M. Wipliez, F. Palumbo, and L. Raffo. Multi-purpose systems: A novel dataflow-based generation and mapping strategy. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3073–3076, 2012. [cited at p. 72]

[65] R.A. Normann and A. Branner. A multichannel, neural interface for the peripheral nervous system. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings*, volume 4, pages 370–375, 1999. [cited at p. 101]

[66] Stuart F Oberman and Michael J Flynn. A variable latency pipelined floating-point adder. In *Euro-Par'96 Parallel Processing*, pages 183–192. Springer, 1996. [cited at p. 89]

[67] F. Palumbo, N. Carta, D. Pani, P. Meloni, and L. Raffo. The multi-dataflow composer tool: generation of on-the-fly reconfigurable platforms. *Journal of Real-Time Image Processing*, pages 1–17, 2012. [cited at p. 72, 86, 89]

[68] Francesca Palumbo, Nicola Carta, and Luigi Raffo. The multi-dataflow composer tool: A runtime reconfigurable hdl platform composer. In *Conf. on Design and Architectures for Signal and Image Proc.*, pages 178–185, 2011. [cited at p. 87]

[69] D. Pani, F. Usai, L. Citi, and L. Raffo. Real-time processing of tfLIFE neural signals on embedded dsp platforms: a case study. In *Proc. 5th International IEEE EMBS Conference on Neural Engineering*, pages 44–47, 2011. [cited at p. 45, 52, 57, 58, 59, 61, 64, 66, 67, 73, 74]

[70] T.M. Parks. *Bounded Scheduling of Process Networks*. Memorandum (University of California, Berkeley. Electronics Research Laboratory). University of California, Berkeley, 1995. [cited at p. 108]

[71] Yevgeny Perelman and Ran Ginosar. An integrated system for multichannel neuronal recording with spike/lfp separation, integrated a/d conversion and threshold detection. *Biomedical Engineering, IEEE Transactions on*, 54(1):130–137, 2007. [cited at p. 102]

[72] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Comput.*, 16(8):1661–1687, August 2004. [cited at p. 50, 52, 59, 60, 61, 77, 94, 113, 114]

[73] S. Radovan, K. Saša, K. Dejan, and D. Goran. Optimization and implementation of the wavelet based algorithms for embedded biomedical signal processing. *Computer Science and Information Systems*, 10:502–523, 2013. [cited at p. 72]

[74] S. Raspopovic, J. Carpaneto, E. Udina, X. Navarro, and S. Micera. On the identification of sensory information from mixed nerves by using single-channel cuff electrodes. *J Neuroeng Rehabil.*, 7(17), April 2010. [cited at p. 2, 59]

[75] R. Rieger, J. Taylor, A. Demosthenous, N. Donaldson, and P. J. Langlois. Design of a low-noise preamplifier for nerve cuff electrode recording. *IEEE Journal of Solid-State Circuits*, 38(8):1373–1379, 2003. [cited at p. 7]

[76] P.M. Rossini, S. Micera, A. Benvenuto, J. Carpaneto, G. Cavallo, L. Citi, C. Cipriani, L. Denaro, V. Denaro, G. Di Pino, F Ferreri, E. Guglielmelli, K.P. Hoffmann, S. Raspopovic, J. Rigosa, L. Rossini, M. Tombini, and P. Dario. Double nerve intraneural interface implant on a human amputee for robotic hand control. *Clin. Neurophysiol.*, (121):777–883, May 2010. [cited at p. 104]

[77] R. Schreier and T. Gabor C. Understanding delta sigma data converters. *IEEE Press/Wiley-Interscience*, 2001. [cited at p. 13, 18]

[78] F. Shahrokhi, K. Abdelhalim, D. Serletis, P. L. Carlen, and R. Genov. The 128-channel fully differential digital integrated neural recording and stimulation interface. *IEEE Trans. Biomedical Circuit System*, 4:149–161, 2010. [cited at p. 21]

[79] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004. [cited at p. 56]

[80] M.J. Shensa. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on Signal Processing*, 40(10):2464–2482, October 1992. [cited at p. 50, 73, 105]

[81] N. Siret, M. Wipliez, J. F Nezan, and A. Rhatay. Hardware code generation from dataflow programs. In *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, pages 113–120, Oct 2010. [cited at p. 88]

[82] M. Tombini, J. Rigosa, F. Zappasodi, C. Porcaro, L. Citi, J. Carpaneto, P.M. Rossini, and S. Micera. Combined analysis of cortical (EEG) and nerve stump signals improves robotic hand control. *Neurorehabilitation and Neural Repair*, 26(3):275–281, 2012. [cited at p. 1]

[83] M. Velliste, S. Perel, M.C. Spalding, A.S. Whitford, and A.B. Schwartz. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453:1098–1101, 2008. [cited at p. 21]

[84] S. Venkatraman, K. Elkabany, J. D. Long, Y. Yao, and J. M. Carmena. A system for neural recording and closed-loop intracortical microstimulation in awake rodents. *IEEE Transaction on Biomedical Engineering*, 56:15–22, 2009. [cited at p. 21]

[85] L. Wang and L. Theogarajan. A micropower delta-sigma modulator based on a self-biased super inverter for neural recording systems. *Custom Integrated Circuits Conference (CICC)*, pages 1–4, 2010. [cited at p. 8]

[86] Y. Wang, Z. Wang, X. Lu, and H. Wang. Fully integrated and low power cmos amplifier for neural signal recording. *IEEE Engineering in Medicine and Biology Society*, pages 5250–5230, 2005. [cited at p. 8]

[87] W. Wattanapanitch and R. Sarpeshkar. A low-power 32-channel digitally programmable neural recording integrated circuit. *IEEE Transaction on Biomedical Engineering*, 5(6):593–602, 2011. [cited at p. 21]

[88] Paul B. Yoo and D.M. Durand. Selective recording of the canine hypoglossal nerve using a multicontact flat interface nerve electrode. *IEEE Transactions on Biomedical Engineering*, 52(8):1461–1469, August 2005. [cited at p. 2]

[89] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C.-S. Poon. Real-time fpga-based multichannel spike sorting using hebbian eigenfilters. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 1(4):502–515, Dec 2011. [cited at p. 102]

[90] A. Zabihian and A.M. Sodagar. A new architecture for multi-channel neural recording microsystems based on delta-sigma modulation. *IEEE Conference on Biomedical Circuits and Systems*, pages 81–84, 2009. [cited at p. 8]

[91] H. Zare-Hoseini, I. Kale, and O. Shoaei. Modeling of switched capacitor delta sigma modulators in simulink. *IEEE Trans. on Instrumental and Measurement*, 54:1646–1654, 2005. [cited at p. 12]

[92] Fei Zhang, Mehdi Aghagolzadeh, and Karim Oweiss. A fully implantable, programmable and multimodal neuroprocessor for wireless, cortically controlled brain-machine interface applications. *J. Signal Process. Syst.*, 69(3):351–361, December 2012. [cited at p. 102]

[93] Ming Zhang, Rangyu Deng, Zhuo Ma, and Minxuan Zhang. A FPGA-based low-cost real-time wavelet packet denoising system. In *Proc. of 2011 Int. Conf. on Electronics and Optoelectronics (ICEOE)*, volume 2, pages V2–350–V2–353, 2011. [cited at p. 73]

[94] Zachary S Zumsteg, Caleb Kemere, Stephen O'Driscoll, Gopal Santhanam, Rizwan E Ahmed, Krishna V Shenoy, and Teresa H Meng. Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 13(3):272–279, 2005. [cited at p. 103]

List of Publications Related to the Thesis

Published papers

Journal papers

- F. Palumbo, N. Carta, D. Pani, P. Meloni, L. Raffo, *The Multi-Dataflow Composer tool: generation of on-the-fly reconfigurable platforms.*, Journal of Real-Time Image Processing (JRTIP), 2012, pp. 1-17. ISSN : 1861-8200, DOI: 10.1007/s11554-012-0284-3. (Relation to Chapter 6)

Conference papers

- F. Palumbo, N. Carta, L. Raffo, *The Multi-Dataflow Composer tool: a Runtime Reconfigurable HDL Platform Composer*, 2011 Conference on Design and Architectures for Signal and Image Processing (DASIP), Tampere, Finland, November 2011, pp. 178-185, ISBN: 978-1-4577-0620-2, **Best Paper Award** (Relation to Chapter 6)

- C. Carboni, N. Carta, M. Barbaro, L. Raffo, *A sigma-delta architecture for recording of peripheral neural signals in prosthetic applications*, Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS and EMBS International Conference on, Rome, Italy, June 2012, pp. 448-453, ISBN: 978-1-4577-1199-2. (Relation to Chapter 2)

- M. Wipliez, N. Siret, N. Carta, F. Palumbo, L. Raffo, *Design IP Faster: Introducing the C High-Level Language*, IP-Embedded System Conference and Exhibition (IP-SOC 2012), Grenoble, France, December 2012. **Best Paper Award** (Relation to Chapter 6)

- N. Carta, C. Sau, F. Palumbo, D. Pani, L. Raffo, *A Coarse-Grained Reconfigurable Wavelet Denoiser exploiting the Multi-Dataflow Composer tool*, 2013 Conference on Design and Architectures for Signal and Image Processing (DASIP), Cagliari, Italy, October 2013, pp. 141-148, ISBN: 979-10-92279-01-6. (Relation to Chapter 6)

- N. Carta, C. Sau, D. Pani, F. Palumbo, L. Raffo, *A Coarse-Grained Reconfigurable Approach for Low-Power Spike Sorting Architectures*, The 6th International IEEE EMBS Neural Engineering Conference (NER 2013), San Diego, California, USA, 6-8 November 2013, pp. 439-442, ISBN:978-1-4673-1969-0. (Relation to Chapter 6)

- N. Carta, D. Pani, L. Raffo, *VLSI Wavelet Denoising of Neural Signals, Critical Appraisal of Different Algorithmic Solutions for Threshold Estimation*, Proceedings of the 7th International Conference on Biomedical Electronics and Devices (BIODEVICES), Angers (France), 3-6 March

2014, pp. 45-52, DOI: 10.5220/0004865700450052. **Best Student Paper Award**. (Relation to Chapter 5)

- C. Carboni, N. Carta, L. Bisoni, M. Barbaro, L. Raffo, *A Bio-Electronic recording module for peripheral neural signals*, National congress of Bioengineering (GNB), Roma, Italy, June 2012, ISBN:978 88 555 3182-5. (Relation to Chapter 2)

- L. Bisoni, C. Carboni, N. Carta, M. Barbaro, L. Raffo, *A Bidirectional Interface to the Peripheral Neural System based on a Sigma-Delta Recording unit and on a high-voltage Stimulator*, National Congress on Electronics (GE 2013), Udine, Italy, June 2013. (Relation to Chapter 3)

Posters with published proceedings

- N. Carta, F. Palumbo, L. Raffo, *Coarse-Grained Reconfigurable Approach for Multi-Dataflow Systems*, Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES 2011), Fiuggi, Italy, July 2011, pp. 97-100, ISBN: 978 90 382 1798-7. (Relation to Chapter 6)

Submitted papers

- D. Pani, N. Carta, L. Citi, S. Raspopovic, S. Micera, L. Raffo, *Real-time neural signals decoding onto off-the-shelf DSP processors for neuroprosthetic applications.* Submitted to the Journal of IEEE Transactions on Biomedical Circuits and Systems (TBCAS). (Relation to Chapter 4)

- N. Carta, P. Meloni, G. Tuveri, D. Pani, L. Raffo, *A custom MPSoC architecture with integrated power management for real-time neural signal decoding.* Submitted to the IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS). (Relation to Chapter 7)

- F. Palumbo, C. Sau, N. Carta, L. Raffo, *Early-Stage Low-power Management of Signal Processing Systems.* Submitted to the Journal of Signal Processing Systems. (Relation to Chapter 6)

- L. Bisoni, N. Carta, R. Puddu, C. Carboni, M. Barbaro and L. Raffo, *A multi-channel recording/stimulation device for neuro-prosthetic application.* Submitted to National congress of Bioengineering (GNB), June 2014. (Relation to Chapter 3)

- C. Carboni, L. Bisoni, N. Carta, M. Barbaro, L. Raffo, *Compact, Multi-Channel, Electronic Interface for PNS Recording and Stimulation.* Submitted to the 17th IASTED International Conference on Robotics Applications, June 2014. (Relation to Chapter 3)