VLSI Wavelet Denoising of Neural Signals
Critical Appraisal of Different Algorithmic Solutions for Threshold Estimation

Nicola Carta, Danilo Pani and Luigi Raffo
DIEE - Dept. of Electrical and Electronic Engineering, University of Cagliari, Via Marengo 3, 09123 Cagliari, Italy

Keywords: Wavelet Denoising, Neural Signal Processing, FPGA, Design Tools.

Abstract: Wavelet denoising represents a common preprocessing step for several biomedical applications exposing low SNR. When the real-time requirements are joined to the fulfilment of area and power minimization for wearable/implantable applications, such as for neuroprosthetic devices, only custom VLSI implementations can be adopted. In this case, every part of the algorithm should be carefully tuned. The usually overlooked part related to threshold estimation is deeply analysed in this paper, in terms of required hardware resources and functionality, exploiting Xilinx System Generator for the design of the architecture and the co-simulation. The analysis reveals 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.

1 INTRODUCTION

Real-time biomedical signal processing aims at extracting the information embedded into a physiological measurement, typically in order to aid the diagnosis, monitor a patient or control an electronic device that, at a certain level, interacts with the patient. 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) (Baig et al., 2013) and electroencephalography (EEG) (Casson et al., 2010) 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 invasive neural signal processing.

The aforementioned requirements can be solved by the development of a low-power Application Specific Integrated Circuit (ASIC). Compared to microprogrammed solutions involving the use of microcontrollers or low-power digital signal processors (DSP), ASIC design requires highly specific skills and a longer development time (Montani et al., 2003), unfortunately leading to non-flexible architectures. 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/ (Nezan et al., 2012) 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 enable rapid prototyping with the possibility of exploiting an alternative language/method rather than HDL in the design phase (Palumbo et al., 2012). 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.

In this paper, Xilinx System Generator has been chosen for its rapid prototyping advantage, evaluating the main performance figures associated to the hardware implementation of a wavelet denoising algorithm used for neural signal processing (Diedrich et al., 2003). The ASIC implementation of this algorithm would be particularly useful in a multichannel neural signal processing context, especially for neural prostheses (Citi et al., 2008) which must be able to work exploiting batteries as power source. 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 (Kuzume et al.,

Carta N., Pani D. and Raffo L..
VLSI Wavelet Denoising of Neural Signals - Critical Appraisal of Different Algorithmic Solutions for Threshold Estimation.
DOI: 10.5220/0004865700450052
In *Proceedings of the International Conference on Biomedical Electronics and Devices* (BIODEVICES-2014), pages 45-52
ISBN: 978-989-758-013-0

2004) 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 is challenged here, revealing how it could lead to very inefficient or even infeasible architectures. The discussion is supported by the results in terms of a preliminary FPGA implementation, evaluating the performance onto an open neural signals database (Quiroga et al., 2004).

2 WAVELET DENOISING

Wavelet denoising is a non-linear filtering technique usually adopted in presence of a Gaussian noise whose spectrum overlaps the useful signal bandwidth. It consists of a sub-band decomposition of the signal, thresholding (introducing the non-linearity) and recomposition. The input signal is decomposed in its low-frequency and high-frequency bands, respectively called "approximation" and "detail". The approximation is split again in the same way repeatedly until the level N of decomposition has been reached. Being the Nyquist frequency of the approximation one half of that of the incoming signal, the sample rate can be reduced (decimated approach) so that the same filters can be used in every level. Alternatively, the sample rate can be preserved upsampling in each level the filter coefficients of the previous one, in the so called *algorithme à trous* scheme (Holschneider et al., 1990). Such a redundant approach leads to the same time resolution in every level and to time invariance (Cohen and Kovacevic, 1996). When the N-th level has been computed, all the details are thresholded either *hard* or *soft*, respectively whether the samples of the detail signals are simply cleared to zero if below the threshold or also the samples above threshold are modified by subtracting the value of the threshold itself.

Recomposition exploits the mirrored version of the corresponding decomposition filters (orthogonal wavelets) or different filters (bi-orthogonal wavelets), every couple of filters taking an approximation and the related thresholded detail, producing an approximation signal by averaging sample-wise their outputs. Only Finite Impulse Response (FIR) filters are used, whose impulse responses depend on the chosen mother wavelet.

2.1 Threshold Estimation

Several methods for calculating the threshold have been presented in literature. The choice of the threshold influences the quality of the denoising so much that even data-specific approaches have been presented so far (Medina et al., 2003). The threshold can be fixed (Kuzume et al., 2004) or adaptive (Radovan et al., 2013), the same or different for all the details. In particular, adaptive thresholds are typically computed estimating the *rms* or the standard deviation σ of the signal at the different levels of the decomposition and then correcting it by a multiplicative factor. Different multiplicative factors have been derived and are preferred by different authors, as for the Minimax (Citi et al., 2008), Stein's Unbiased Risk (Mahmoud et al., 2008) or Universal (Bahoura and Ezzaidi, 2012) 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:

$$MAD = median_i\left(|X_i - median_j(X_j)|\right) \quad (1)$$

Due 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:

$$\overline{MAD} = median_j\left(|X_j|\right) \quad (2)$$

It has been proved 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 (SNR). 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 biomedical signal processing (Pani et al., 2011; Zhang et al., 2011). In the following sections, the adoption of the MAD and the sample standard deviation on a sliding window is 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 (Citi et al., 2008), using as a testbed a prototypical FPGA implementation programmed exploiting the Xilinx System Generator tool.

3 METHODS

As starting point, we consider a stationary wavelet denoising trellis ("*algorithme à trous*"). In order to keep low the memory requirements in the perspective FPGA implementation, the simple Haar wavelet has been chosen, requiring very small filters. Considering an input signal sampled at 12kHz as in (Pani et al.,

2011), with 4 levels and removing the last approximation signal, a high-pass overall behaviour able to reject the low-frequency 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 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 we already stated above, 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, as we will discuss hereafter. In this exemplary application, we consider two alternative solutions 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:

$$\theta = \frac{\overline{MAD}}{0.6745}\sqrt{2\log M} \tag{3}$$

in the first case, and:

$$\theta = \sigma\sqrt{2\log M} \tag{4}$$

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, 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. Starting from the technique used in (Pani et al., 2011) 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:

$$s_j = \sum_{n=1}^{N} d_j^2[n] \tag{5}$$

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

$$\sigma = \sqrt{\frac{1}{4N-1}\sum_{k=1}^{4} s_j} \tag{6}$$

Thanks to the sliding window approach, the threshold value is updated every N sampling periods ($M = 4 \times N$). The longer the observation window, the better the estimation accuracy, provided that instantaneous variations (neural spikes) do not influence the threshold computation. Such a processing can be delegated to a host processor picking up the detail signals samples at the 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.

3.1 Architecture Design using Xilinx System Generator

In order to evaluate the different solutions from a 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 of 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.

The tools allow 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.

The wavelet denoising algorithm, created exploiting the Xilinx System Generator tool, is shown in Fig. 1. 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 system period, the target board to map the hardware sub-model, and so on. In our tests, a Xilinx Virtex-5 LX330 has been chosen for its considerable

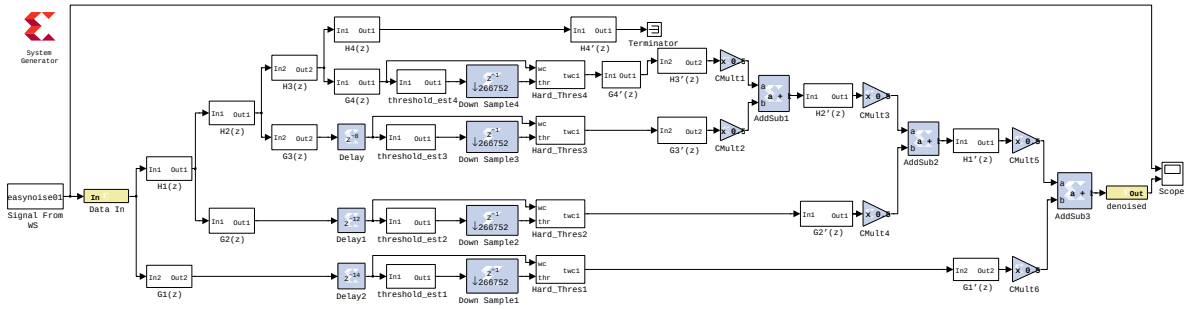


Figure 1: Simulink model of the chosen wavelet denoising scheme using System Generator blocks.

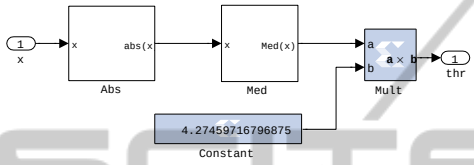


Figure 2: The Simulink Model of the Threshold Estimator block with the constant defined as for the case of M=64 samples per window.

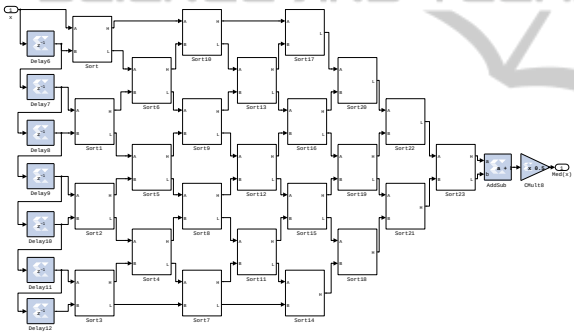


Figure 3: The Simulink Model of the sorter using an unfolded combinatorial approach for windows with M=8 samples.

amount of available resources.

3.2 Threshold Estimator Implementation

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 (3). The corresponding Simulink model is depicted in Fig. 2. The median value calculation requires the hardware implementation of a sorting algorithm which represents a costly operation from a hardware point of view.

A first possible solution can be the unfolded sorter presented in (Bahoura and Ezzaidi, 2012), for which the Simulink model considering windows of $N = 8$ input samples is presented in Fig. 3. 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 (Martinez et al., 2005) 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 4 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

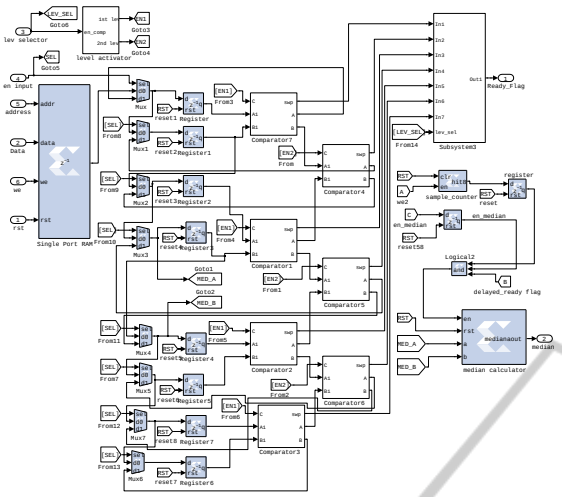


Figure 4: The Simulink Model of the sorter using an iterative approach for windows with M=8 samples.

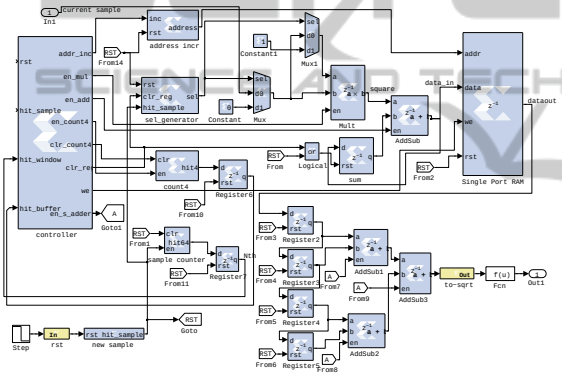


Figure 5: The Simulink Model of the Threshold Estimator block based on the calculation of the sample standard deviation.

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. Figure 5 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.

4 EXPERIMENTAL RESULTS

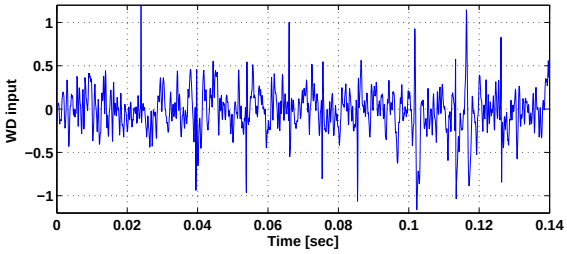
Before analysing the results in terms of hardware resources which are necessary for the different solutions presented above, such solutions have been evaluated from a functional perspective. A publicly available dataset of simulated neural signals obtained from real physiological action potentials recorded from animals at the central nervous system level has been used (Quiroga et al., 2004). 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 (of tunable intensity) of the neurons at a greater distance from the recording electrodes. The sampling frequency has been scaled to 12kHz and the useful bandwidth is declared to be in the range 300Hz - 3kHz.

4.1 Functional Evaluation

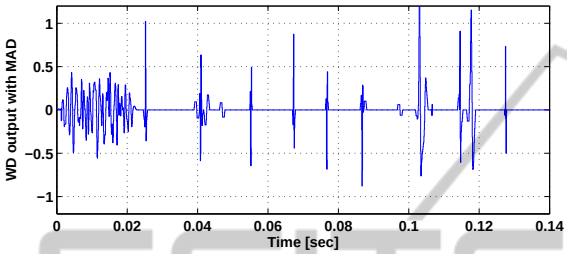
The different versions of the whole wavelet denoising system have been mapped on the target FPGA in order to evaluate, by hardware-software co-simulations, the system performance under real conditions. Figure 6 shows in the first row the neural signal with a low level of background noise used as input 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 significant 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. 7. Even using neural signals with very low SNR, the two implementations behave similarly preserving the relevant spikes.

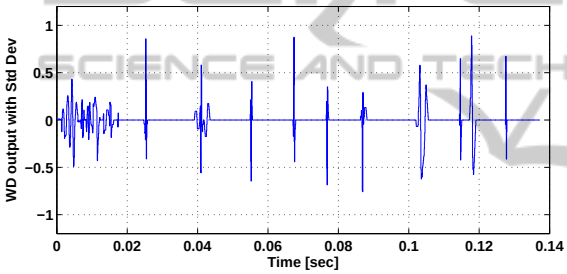
Then, we considered the possibility of enlarging the observation window in order to provide a more significant frame for computing the statistics on the signal. For example, Fig. 8 shows the outputs of the



(a) Input Signal

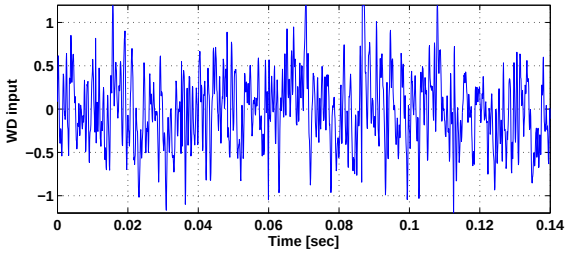


(b) WD output using the $\overline{MAD}$-based threshold

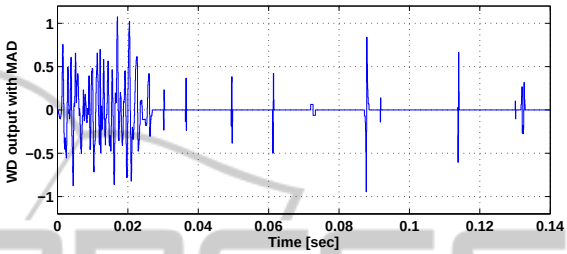


(c) WD output using the σ-based threshold

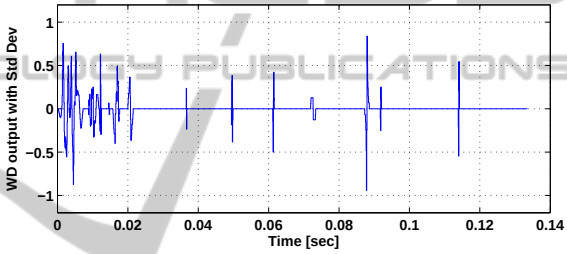
Figure 6: Wavelet Denoising input and outputs: low level noise, N=64 samples.



(a) Input Signal



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



(c) WD output using the σ-based threshold

Figure 7: Wavelet Denoising input and outputs: high level noise, N=64 samples.

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.

We 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 a good denoising.

As can be noticed from Fig. 9, 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.

4.2 Hardware Figures of Merit

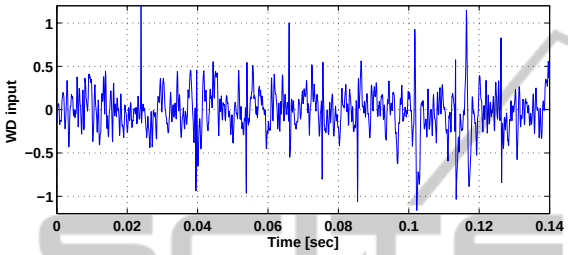
Thanks to the possibilities offered by Xilinx System Generator, also to map the implemented designs on real hardware, in this case the Xilinx FPGA Virtex-5 LX330 device, we evaluated pros and cons in terms of necessary hardware resources for the different solutions presented above. The final goal is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

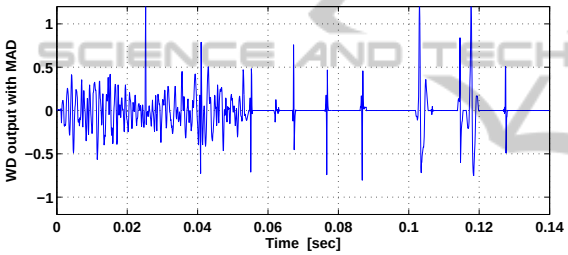
The solution based on the combinatorial (unfolded) $\overline{MAD}$ implementation, as highlighted in Table 1, is absolutely inefficient, taking into account it

Table 1: FPGA synthesis results for the *Threshold Estimator* varying the length of the observation window.

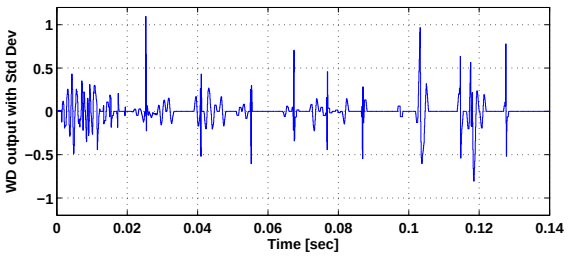
	N	$f_{max}[MHz]$	Slice Registers	LUTs
σ	32	417.34	156 / 207360 (0.07%)	80 / 207360 (0.04%)
	64	416.61	157 / 207360 (0.07%)	82 / 207360 (0.04%)
	128	416.02	160 / 207360 (0.07%)	84 / 207360 (0.04%)
unfolded $\overline{MAD}$	8	417.08	558 / 207360 (0.27%)	20270 / 207360 (9.77%)
folded $\overline{MAD}$	32	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%)



(a) Input Signal



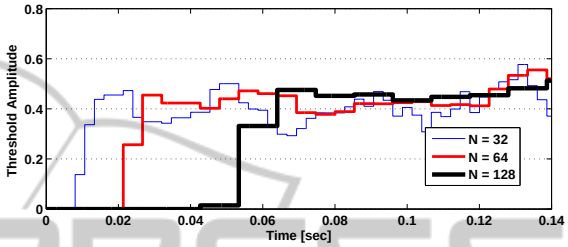
(b) WD output using the $\overline{MAD}$-based threshold



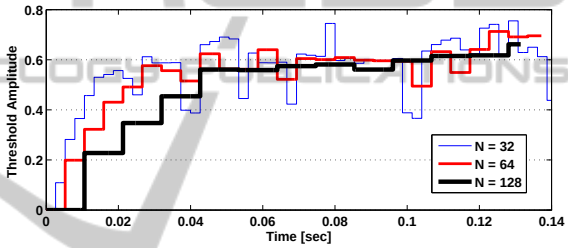
(c) WD output using the σ-based threshold

Figure 8: Wavelet Denoising input and outputs: low level noise, N=128 samples.

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



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



(b) σ-based solution

Figure 9: Threshold variation over time using different lengths of the observation window.

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.

5 CONCLUSIONS

This paper highlights how 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 $\overline{MAD}$ reveals similar func-

tional 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. The paper also stresses the benefit of using hardware-software co-simulations tools such as Xilinx System Generator for rapid prototyping and verification on FPGA, which represents a value added for the research in rapidly evolving fields such as neural engineering, overcoming the limits of the traditional HDL coding.

ACKNOWLEDGEMENTS

The research leading to these results has received funding by the Region of Sardinia in the ELoRA project (Fundamental Research Programme, L.R. 7/2007, grant agreement CRP-60544), by the European Commission in the NEBIAS project (FP7, FET Proactive, grant agreement 611687), by Italian Government in the HANDBOT project (PRIN 2010/11, prot. 2010YF2RY_003) and by the Italian Ministry of Health in the NEMESIS project (Young Researchers, grant agreement 064/GR-2009-1591615).

REFERENCES

Bahoura, M. and Ezzaidi, H. (2012). FPGA-implementation of discrete wavelet transform with application to signal denoising. *Circuits, Systems, and Signal Processing*, 31(3):987–1015.

Baig, M. M., Gholamhosseini, H., and Connolly, M. J. (2013). A comprehensive survey of wearable and wireless ECG monitoring systems for older adults. *Medical & Biological Engineering & Computing*, 51(5):485–495.

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

Citi, L., Carpaneto, J., Yoshida, K., Hoffmann, K.-P., Koch, K. P., Dario, P., and Micera, S. (2008). 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.

Cohen, A. and Kovacevic, J. (1996). Wavelets: the mathematical background. *Proceedings of the IEEE*, 84(4):514–522.

Diedrich, A., Charoensuk, W., Brychta, R., Ertl, A., and Shiavi, R. (2003). 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.

Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer Berlin Heidelberg.

Kuzume, K., Niijima, K., and Takano, S. (2004). FPGA-based lifting wavelet processor for real-time signal detection. *Signal Processing*, 84(10):1931–1940.

Mahmoud, M. I., Dessouky, M. I. M., Deyab, S., and Elfouly, F. H. (2008). Signal denoising by wavelet packet transform on FPGA technology. *Special Issue of Ubiquitous Computing and Communication Journal Bioinformatics and Image*.

Martinez, J., Cumplido, R., and Feregrino, C. (2005). An FPGA-based parallel sorting architecture for the Burrows Wheeler transform. In *International Conference on Reconfigurable Computing and FPGAs, ReConFig 2005*.

Medina, C., Alcaim, A., and Jr., J. A. (2003). Wavelet denoising of speech using neural networks for threshold selection. *Electronics Letters*, 39(25):1869–1871.

Montani, M., Marchi, L. D., Marcianesi, A., and Speciale, N. (2003). 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.

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

Palumbo, F., Carta, N., Pani, D., Meloni, P., and Raffo, L. (2012). The multi-dataflow composer tool: generation of on-the-fly reconfigurable platforms. *Journal of Real-Time Image Processing*, pages 1–17.

Pani, D., Usai, F., Citi, L., and Raffo, L. (2011). Impact of the approximated on-line centering and whitening in OL-JADE on the quality of the estimated fetal ecg. In *Proc. of the 5th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 44–47.

Quiroga, R. Q., Nadasdy, Z., and Ben-Shaul, Y. (2004). Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Comput.*, 16(8):1661–1687.

Radovan, S., Saša, K., Dejan, K., and Goran, D. (2013). Optimization and implementation of the wavelet based algorithms for embedded biomedical signal processing. *Computer Science and Information Systems*, 10:502–523.

Zhang, M., Deng, R., Ma, Z., and Zhang, M. (2011). 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.