Continuous Time Neural Signal Processing in Embedded Platforms

João Lovegrove Pereira
joao.lovegrove.pereira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2016

Abstract

When recording the electrical activity of the brain at an extracellular level, a process of spike sorting is necessary to classify each detected action potential to its source neuron, attaining single-neuron resolution. Current trends have pushed spike sorting to be performed on an implant, invasively placed with the electrodes. This requires efficient spike sorting algorithms to minimize power consumptions and chip sizes. In this work I study the possibility of continuous time data conversion and asynchronous spike sorting improving the efficiency of such implants without loss of sorting performances. I find that, due to the spurious behaviour of spikes, fewer samples are produced by the event-driven sampling scheme of level-crossing. From the continuous time outputs of level-crossing sampling, I propose a set of 4 simple features that can be economically extracted, while attaining similar performances to template matching, a reference spike sorting method. The work finishes with the demonstration of the proposed spike sorting system implemented on an FPGA. Although a conclusive comparison between this new paradigm and conventional systems will depend greatly on the final hardware implementation, the results reveal the potential of harnessing the spurious behaviour of extracellular recordings to reduce costs.

Keywords: Spike Sorting, Continuous Time, Level-Crossing, Event Driven, Asynchronous, FPGA

1. Introduction

The brain is composed of billions of neurons that together perform complex functions such as memory and cognition, which despite ongoing efforts by the scientific community, are still not completely understood. The main drive behind these efforts is the desire to develop ways to prevent, manage or cure the many neurological conditions that can occur with a possible great impact on people's lives.

With technological advances in electronics and robotics it's also becoming possible to interface the brain directly with sensory and effector units to restore lost functions from spinal cord injuries, brainstem stroke or other disorders. Examples of such brain-machine interfaces, BMIs, are cochlear implants and neural prostheses. Furthermore, understanding the brain can also inspire the development of novel technologies such as artificial neural networks and neuromorphic electronic architectures.

A key requirement to further our knowledge of the brain is the ability to record the electrical activity of individual neurons, i.e. achieve single-neuron resolution. This is done by implanting microelectrodes into the neural tissue to capture the extracellular action potentials. As many neurons might be recorded simultaneously, a spike sorting algorithm is necessary to classify each to its source neuron.

Advances in semiconductor technology are enabling integrating these algorithms on application-specific integrated circuits, ASICs, to be implanted with the electrodes. However, faced with the desire to record ever increasing numbers of channels simultaneously, there is a demand for efficient spike sorting algorithms, that satisfy the strict power and area constraints of such implants.

This work raises the hypothesis of continuous time, CT, data conversion and event-driven spike sorting enabling lower power consumptions by:

- producing fewer, more informative samples through level-crossing, LC, sampling;

- asynchronous delta-modulation enabling economical feature extraction;

- the activity-dependent dynamic power consumption of asynchronous digital signal processors, DSPs, allowing power savings when applied to sparse signals.

This paper is organized as follows. Section 2 explains in further detail the process of spike sorting and its several stages, finishing with a review of the state-of-the-art of on-chip spike sorting sys-

tems. This is followed by the developed work, which is divided into 3 parts.

Section 3 explores the use of CT data conversion on extracellular recordings. It begins by explaining LC sampling and its properties. A LC analogue-to-digital converter, ADC, model created on *Matlab* is then applied to artificial neural datasets with the objective of determining under which conditions its use is beneficial compared with conventional uniform sampling, US.

Section 4 proposes 4 spike sorting features that can be economically extracted from the CT output of LC ADCs, and evaluates how these perform, compared with a reference spike sorting method, in terms of sorting performances and computational cost.

Finally in Section 5 the proposed system is implemented on an FPGA as a proof-of-concept of how the CT output of LC ADCs can be asynchronously processed for spike sorting. The paper finishes with rounding-up conclusions and suggestions for further work, in Section 6.

2. Background

An action potential is a characteristic bioelectrical event that can occur in neurons. It consists on a sharp increase of the membrane's potential followed by a return to the resting value, usually lasting 1 to 2 milliseconds. Action potentials play a crucial role in neural communication, enabling the brain's computational abilities. Their shape doesn't change as they propagate down each neuron. Rather it's the number and timing between action potentials that is thought to carry information.

The electrical activity of the brain can be recorded at different levels of invasiveness using electrodes, though to distinguish the activity of individual neurons, i.e. to obtain single-neuron resolution, microelectrodes must be placed directly into the neural tissue.

Single-neuron resolution is necessary to understand how complex phenomena such as neural coding and plasticity occur, enabling a bottom-up approach to understand the higher-level functions of the brain. From an applications point of view it can, for example, improve decoding of intentions in BMIs [1].

An extracellular recording is composed of various contributions. Local field potentials consist on low frequency oscillations of the baseline due to the cumulative activity of populations of neurons away from the electrode. The recording also contains noise, which can have a biological source, or result from the recording instrumentation, for example the thermal noise added from the relatively high impedance of electrodes.

Our signal of interest however, are spikes. These consist on extracellular action potentials, that have peaks ranging $50 - 500\ \mu V$.

2.1 Spike Sorting

Each extracellular recording might contain spikes from several different neurons, located close to the electrode's tip. To obtain the desired single-neuron resolution, spike sorting is required. It is based on 2 assumptions:

- The typical spike waveform of each neuron doesn't change throughout a recording;

- Spikes from different neurons will be recorded having different waveforms.

The latter results from a series of factors. Different neuron geometries, and concentrations of membrane ionic channels, result in different shapes of action potentials [2]. The recorded spike's amplitude also decreases as the distance between the electrode and source neuron increases. These differences are used to distinguish the spikes of different neurons.

Neural recordings are usually performed with the electrodes in the brain and wires relaying the data outside to acquisition systems. The use of transcutaneous wires, however, is not ideal, due to the risk of infections, in addition to restricting the movement of the subjects. Thus, a wireless data transmitter can be included with the electrodes, bypassing the need for wires.

Due to the relatively high frequency of spikes and the fine resolutions needed to capture their details, high amounts of data can easily be produced. For example, a system with 128 channels can require datarates up to 23 Mbps. Additionally, in the past years the number of electrodes used in each recording has grown exponentially, doubling approximately every 7 years [3], with some of today's recordings using hundreds simultaneously.

On the other hand, neural implants are constrained by their overall size, which needs to be kept to a minimum, to reduce trauma from implantation. It's dissipated power mustn't also cause heat-induced damage to biological tissues, with the literature stating a maximum allowed temperature increase of $1°C$ [4].

These high bandwidths, together with the strict design constraints, create a bottleneck with current channel counts lagging in comparison with microelectrode fabrication advances.

A solution is to perform data reduction prior to transmission. With on-chip spike sorting, the whole recording can be reduced to binary pulses indicating when a spike fires and the corresponding class. As this is already the desired input for many applications, no relevant information is lost. Datarate reductions of $200\times$, compared with transmitting the raw data, can be achieved.

Several stages are implemented on-chip for spike sorting. The analogue-front-end, AFE, provides an

interface between the electrodes and the DSP. It includes a low-noise amplifier, with typical amplification gains of 50-200 [5], and a bandpass filter, to remove the low frequency local field potentials and high frequency noise. The highpass cutoff frequency is usually 100-300 Hz and lowpass frequency is 2-6 kHz. The AFE also includes an ADC, with typical sampling rates, SR, of 16-32 kHz and 8-10 bits for amplitude quantization. These parameters define the accuracy and computational cost of the following stages.

Spike detection results in a window of data, usually around 2 ms long, considered to contain a spike. Most methods consist on applying a threshold to the neural data, either directly or after an operator has been applied to emphasize spikes from the background noise. Spikes can be aligned to the detection or another reference point, such as their maximum, though this will require using a data buffer to include in the data window samples that occurred prior to detection, incurring additional costs for this stage.

From each detection window, a set of features is extracted, which aims at succinctly describing the spikes. The most straightforward method is to use the samples directly, which consists on the spike sorting method of template matching, TM. Extracted features can relate with the waveform of spikes, for example, corresponding to their peaks or derivatives, or be obtained through analytical methods such as principal component analysis, PCA, or discrete wavelet transform, DWT.

As spikes need to be classified in real-time, the computational complexity for spike sorting is restricted. Thus, classification is usually divided into two phases.

In the training phase, clustering is done to find the underlying number of neurons and their corresponding waveforms. Different methods can be used, such as k-means and expectation maximization. For a detailed review of sorting algorithms refer to [6]. Motivated by their computational intensity, some systems implement this stage offline. In the classification stage, a computationally lighter classifier is used, based on the training results. It can use, for example, classify spikes to the class of the closest centroid, based on a distance metric such as the Euclidean or l1-norm. Alternatively, the feature space can be divided into regions, and spikes be classified by comparison with the boundaries.

2.2 State-of-the-Art

A few on-chip spike sorting solutions have been reported so far, though in practice most recordings still rely on wires to transmit the data.

In [7] an FPGA was used for spike sorting, with detection by amplitude threshold crossing. Spikes are first transmitted to a computer where training is done. The mean waveform of each class is then sent back to the FPGA where real-time spike sorting is achieved through TM. The system, however, is not implantable, being intended for a headstage for animal experiments.

In [8] a similar 2-stage approach was implemented on a microcontroller, MCU. From a group of possible features, consisting on the samples and the derivatives of each spike, an optimized subset is found on a computer. Real-time classification is then performed on the MCU by comparing each optimized feature with a threshold, also found offline, to cast a vote, with the majority determining the class. This performance-oriented reduction of the feature set was shown to result in better sorting performances, while having lower costs, than other methods such as PCA.

Although using off-the-shelf components has the advantage of easier and faster development, ASICs ultimately enable lower power consumptions and implant sizes.

In [9] a fully analogue IC performed spike detection by amplitude thresholding both the positive and negative peaks of spikes. It then extracted the maximum, minimum and width of the main peak as features. Although spike sorting was not included on-chip, using real datasets from rats and monkeys, the proposed features were found to achieve 90 % of the accuracy of PCA.

In [10] a digital 64-channel spike detector and feature extractor was implemented. Detection was done using the non-linear energy operator on the signal, which results in amplification where both amplitude and derivative are high. Spikes were aligned to the maximum of their derivative, and the derivatives were used as features. Using simulated datasets with signal-to-noise ratios, SNRs, raging from -15 to 20 dB, the sorting accuracy using these features was of 77%. The IC had a SR of 24 kHz and used 8 bits for amplitude quantization.

To the extent of my knowledge, the only spike sorting IC with on-chip unsupervised clustering was presented in [11]. Detection was done by absolute value thresholding, and TM was used for sorting. The clustering algorithm implemented consists on an adaptation of OSort, an online method presented in [12]. Efficient resource utilization is done by training each channel sequentially. If necessary, training can be repeated, rendering the system adaptable. The system achieved 75% of sorting accuracy using simulated datasets with a SNR raging from 0 to 15 dB. The clustering performance was comparable to offline methods, such as k-means, in spite of the disadvantage of not having all the data at the beginning of clustering.

The works cited in this brief review are summarized in Table 1. Spike sorting results are not in-

Table 1: State-of-the-art of the neural spike sorting integrated systems reviewed.

Reference	[7]	[8]	[9]	[10]	[11]
Year	2015	2016	2008	2011	2013
Technology	FPGA	MCU	350 nm	90 nm	65 nm
Area/Channel (mm^2)	/	/	0.24	0.06	0.08
Power/Channel (μW)	1000	268	70	2	4.7
On-Chip Training	No	No	No	No	Yes
On-Chip Sorting	Yes	Yes	No	No	Yes

cluded due to the variety of ways these are presented, and types of datasets used.

3. Continuous Time Data Conversion

Conventionally, digital signal processing circuits are synchronous. A clock is used to define the SR of the ADC and to govern the switching of the digital circuitry.

When sampling extracellular recordings, high SRs are needed to capture the detail of spikes' waveforms. However, the signal shows a spurious behaviour, with high frequency spikes intercalated by periods of lower activity. This sparsity results in many of the samples produced by uniform sampling, US, not carrying any information on spikes, thus wasting power without benefits to spike sorting.

This has motivated studying CT data conversion to improve the efficiency of spike sorting implants. With a sampling scheme such as LC sampling, samples is only produced when events of interest occurs in the input signal. As this can happen at any moment, they are spread continuously in time. The SR is thus data-driven, adapting to the activity of the input. If smooth, few samples are produced, but as the activity increases, so does the SR.

3.1 Level-Crossing Sampling

For LC sampling the input analogue signal is compared with a set of pre-defined amplitude thresholds. If either is crossed, a sample is produced. The difference between these is always 1 least-significant-bit, lsb, though samples will not be equally separated in time, as with US.

In practice, the input signal only needs to be compared with the 2 thresholds directly encompassing it, which can be implemented with comparators acting as level-crossing detectors. These are connected to a counter which increments or decrements by 1 if the upper or lower threshold is crossed respectively. The value of the counter corresponds to the digital sample, which is subtracted to the input analogue signal, using a feedback digital-to-analogue converter, DAC. Thus, the comparison thresholds can be fixed at $\pm$ 1 lsb. Note how neither sample&hold nor a clock signal are needed.

Instead of producing samples in the conventional sense, the value of the counter can be streamed directly in the DSP. I will continue, however, to refer to changes to the counter's value as if a sample was produced.

The most relevant property for spike sorting is the data-driven SR. While the designer of US ADCs can define both the SR and amplitude resolution, r, with LC ADCs, only the latter is fixed:

$$r = \frac{V_{FS}}{2^N} \tag{1}$$

V_{FS} is the maximum input range and N is the number of bits used for amplitude quantization. The SR will result from the relation between r and the input signal's gradient and amplitude. If set too narrow, too many samples might be produced on the slopes of spikes. A balance is thus required between the desired amplitude resolution and the number of samples produced. For a sinusoidal input of amplitude A and frequency f, the average number of samples per second is:

$$N_S = 4\frac{A}{r}f \tag{2}$$

A possible weakness of LC sampling is noise. If too high, it can cause erroneous level-crossings, increasing the number of samples without adding information. For a Gaussian noise input signal with bandwidth f_m and power φ, the average number of samples produced per second is [13]:

$$\overline{N_S} = \sqrt{\frac{8\pi}{3}}\frac{\sqrt{\varphi}}{r}f_m \tag{3}$$

For this reason, an adequate AFE filter will be essential.

Although in theory LC ADCs will not produce aliasing, they still have a maximum input frequency beyond which saturation error occurs, i.e. a delay between the counter's value and the input signal. It results from the loop delay of the ADC, Δ, which represents the time it takes to update the counter, DAC and comparators after each level-crossing, before being able to process the following one. The maximum input gradient to be sampled without saturation error is:

$$\left|\frac{dV_i}{dt}\right| < \frac{r}{\Delta} \tag{4}$$

Which corresponds to a maximum loop delay for a full scale sinusoidal input of:

$$\Delta = \frac{1}{2^N \pi f_{in^{max}}} \tag{5}$$

ADC State-of-the-Art: LC vs US

Different US ADC architectures have been developed for optimal power consumptions at different SRs and resolutions. For electrophysiological signals, SAR ADCs are the most appropriate. They have a similar architecture to LC ADCs, using a feedback DAC, though only 1 comparator is needed. To produce each N-bit sample, N comparisons are required to do a binary search. Thus, if sampling at 16 kHz and using 8 bits, the internal operating frequency of these ADCs is 128 kHz. LC ADCs on the other hand require only one step per sample. Additionally, their DAC only suffers voltage swings of 1 lsb, while for SAR ADCs they're required to settle to the full range of the ADC with each step.

Two common metrics are used to describe the performance of ADCs. The signal-to-noise and distortion ratio, SINAD, is measured from the spectrum of the output as:

$$SINAD = \frac{P_{signal} + P_{noise} + P_{distortion}}{P_{noise} + P_{distortion}} \tag{6}$$

The effective number of bits, ENOB, corresponds to the number of bits an ideal ADC with the same SINAD would need:

$$ENOB = \frac{SINAD - 1.76}{6.02} \tag{7}$$

State-of-the-art SAR and LC ADCs are compared in Table 2. Lower power consumptions are reported for the SAR ADC, with the 2 comparators of the LC ADC accounting for approximately 57% of its total power consumption. The SAR ADC is also less restricted by the maximum input frequency. Nonetheless, the possibility of LC ADCs producing fewer samples could result in overall lower power consumptions. Additionally, it's reasonable to expect further improvements to LC ADCs, as they are relatively recent and have an architecture similar to SAR ADCs.

Asynchronous Delta-Modulation

A further advantage of LC ADCs is asynchronous delta-modulation. Proposed in 1966 [13] it consists on representing a signal by a succession of binary pulses indicating whenever it changes by a fixed amount. In other words, whenever the upper or lower thresholds of the LC ADC have been crossed, an INC or DEC pulse is created respectively.

Table 2: Comparison between state-of-the-art LC and SAR ADCs.

	LC	SAR
Reference	[14]	[15]
Year	2014	2015
Technology (nm)	130	180
Area (mm^2)	0.28	0.16
Power (μW)	0.21	0.04
Supply (V)	0,3	0,6
SINAD (dB)	28	58
ENOB	4,4	9,4
SR	/	20 kHz
Max F_{in}	1.2 kHz	10 kHz

To facilitate the control of the following DSP stages, explained in Section 4, the INC and DEC pulses can be transformed, through some digital logic, into a request, REQ, and a direction, DIR, pulses, with REQ indicating whenever a level-crossing occurs, and DIR indicating if it was an INC or a DEC.

3.2 Application to Neural Data

To determine how LC sampling performs on extracellular recordings, a *pseudo*-CT LC ADC was implemented on *Matlab*. The difference between consecutive thresholds, *thrID*, is found as:

$$thrID = \frac{max(data) - min(data)}{2^N} \tag{8}$$

With N the number of bits used for amplitude quantization. To simulate CT, the datasets were upsampled to 10 MHz, using linear interpolation between consecutive samples.

16 artificial datasets were used in this work, which have the advantage of including ground-truth knowledge, i.e. the timing and true class of each spike. The datasets were created in [16] and made publicly available. From a database of 594 different spike waveforms from monkey neocortex and basal ganglia, 3 different spike shapes are used in each dataset to represent 3 different source neurons, each with a mean firing rate of 20 Hz. To generate background noise, spikes randomly selected from the whole database are added at random times and amplitudes.

The 16 datasets are divided in half as 'Easy' or 'Difficult', given the similarity between their spikes waveforms, and have 4 different noise levels, 0.05, 0.1, 0.15 and 0.2, computed as:

$$Noise_{level} = median\left\{\frac{|signal|}{0.6745}\right\} \tag{9}$$

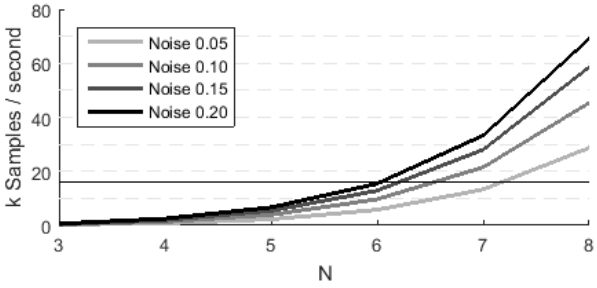

Figure 1: Average number of samples per second produced by LC sampling neural recordings. The horizontal line marks the number of samples per second produced by US at 16 kHz.

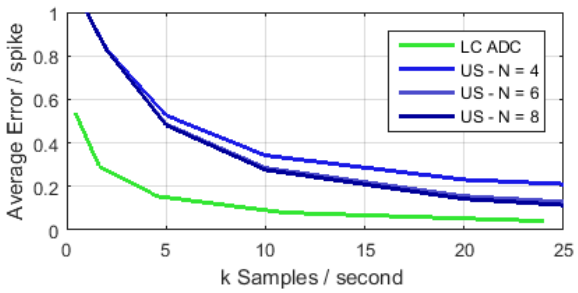

Figure 2: Digitization error as a function of the number of samples per second produced by LC and US. For LC sampling N ranges from 3 to 8. The value of the error is normalized to the maximum.

All datasets were filtered using a 4^{th} order elliptic filter implemented on *Matlab*. The lowpass cutoff frequency was set to 4000 Hz and the high pass cutoff frequency to 200 Hz.

Results

To observe how N influences the number of samples produced on neural recordings, 10 seconds were used for each dataset, Figure 1 a). It can be seen how the average number of samples displays an exponential growth with N, as was expected from Eq. 2. For N larger than 7, a prohibitively high number of samples is produced, while if lower than 6, they are much fewer compared with the typical uniform sampling rate of 16 kHz.

As N increases, the maximum loop delay allowed decreases exponentially, as expected to Eq. 5. It is found, for example, that for $N = 6$, $\Delta < 2.5$ μs.

We are concerned not only with the number of samples, but also with their quality. For this, we isolated the spikes using the ground-truth, then sampled them using both LC and US. The digitized signals were reconstructed using zero-order interpolation and compared with the original spikes by computing the error:

$$E = \frac{1}{\#spikes} \sum_{i}^{\#spikes} |spike^{i}_{rec} - spike^{i}| \qquad (10)$$

The results can be seen in Figure 2, with the error being a function of the average number of samples per second produced. For LC sampling this depends solely on N, while for US it depends on the SR. Thus three curves are shown for different amplitude resolutions.

As the number of samples increases, both methods tend to the same error, though for lower numbers of samples, LC sampling proves to be more informative due to its event-driven nature.

This approach, however, is limited to N lower than 7, which could cripple sorting results. Also, it raises difficulties at the DSP stage, namely, how the CT outputs of these ADCs, the REQ and DIR pulses and the counter's value, can be efficiently used for spike sorting.

4. Asynchronous Spike Sorting

The CT nature of LC ADCs motivates the use of asynchronous digital circuits for spike sorting. With these, the switching frequency of transistors will adapt to the activity of the input signal, resulting in adaptive dynamic power consumptions. A clock to set the SR and govern the DSP is also not required, enabling savings by removing clock drivers, which can also improve modularity.

Electromagnetic interferences in asynchronous circuits are also less severe, being more evenly distributed, as opposed to concentrated at the frequency band of the clock and its harmonics. And less stress on the power distribution network is possible, as power is also drawn more evenly in time.

This approach, however, raises challenges such as the difficulty in circuit design to manage the control pulses and avoid race conditions. Additionally, the timing information between the outputs of LC ADCs contains information, unlike US.

Some CT DSP have been implemented [17], which preserve this timing information using delay lines, to implement CT digital FIR filters. Due to their cost, however, in this study we have opted for explicit time quantization whenever needed, which can be done with an oscillator and a counter. This is the assumption in this study, though the best technology for implementing it has not been explored.

4.1 Event-Driven Features

From the outputs of the LC ADC, we propose extracting a set of 4 event-driven features, EDF:

- Extrema Values - *max* and *min*;

- Minimum Inverse Derivatives - *minDInc* and *minDDec*.

Although simple, they combine the advantages of waveform and derivative features, which are re-

spectively more robust against high and low frequency noise. They can also be extracted as the level-crossings occur, reducing the amount of memory required.

Extrema

In conventional synchronous systems, every sample needs to be inspected to find the extrema. However, we know these only occur at inversion points. With LC sampling, by analysing the REQ and DIR pulses, these can be economically located, reducing the number of comparisons needed to find these features. If a DEC follows an INC, we have a local maximum, and if an INC follows a DEC we have a local minimum. No time quantization is required.

Minimum Inverse Derivatives

In synchronous digital systems the derivatives are found by simply subtracting consecutive samples. With LC sampling, however, the time between samples is not constant. But on the other hand, their difference in value is always 1 lsb. Thus, an inverse derivative can be as:

$$\frac{\Delta t_i}{\Delta x} = \frac{t_i - t_{i-1}}{x_i - x_{i-1}} = t_i - t_{i-1} \tag{11}$$

If the signal has a sharp increase, the inverse derivative will be small. For smooth signals, the time between samples will be larger. A distinction can be made between the inverse derivative of consecutive INC and consecutive DEC pulses. The minima of these are the features: *minDInc* and *minDDec*.

This is the only stage where time quantization will be needed. As we are only interested on the minima of the inverse derivatives, it can be performed only over a short interval, outside of which the pulses in question do not produce minima.

4.2 Methodology

To analyse how the proposed EDF perform in spike sorting, we used 20 seconds of each dataset and the *Matlab* LC ADC described in Section 3.

Amplitude threshold crossing is used for detection, thus, only the instantaneous value of the input is required, avoiding systematic time quantization. Spikes are also aligned to detection point, reducing memory requirements. Using the ground-truth knowledge of the datasets, perfect detection is done, rejecting false positives and overlapping spikes, for better interpretation of the results.

With each detection, features are extracted from the following 1 ms long window. Perfect training is also done, computing the centroids of each class as the median of the spikes belonging to that class. The median was used as it resulted in better performances than the average. Spikes are then classified to the closest centroid using the l1-norm:

$$\begin{aligned} &Class_j = min_i ||Features_j - Centroid_i||_1 \\ &i = 1, 2, 3 \ ; \ j = 1, ..., \#spikes \end{aligned} \tag{12}$$

To evaluate performances we compute the accuracy of sorting as:

$$Accuracy_i = \frac{TP_i + TN_i}{Total} \tag{13}$$

With TP_i the number of spikes correctly classified to class i, TN_i the number of spikes neither belonging nor classified to class i and $Total$ corresponding to the total number of spikes. The accuracy results were averaged among the three spike classes.

To compare the results with a reference method, TM was performed on the same spikes using the same methodology.

Computational cost is reported by the number of operations required, considering 1 operation as a 1-bit addition, subtraction or comparison, or a modulus operation.

To extract *max* and *min* we need one 1-bit comparison per event (level-crossing) to find the inversion points, plus one N-bit comparison per inversion point to determine the extrema:

$$Cost_{max+min} = \overline{\#events} + \overline{\#inv.points} \times N \tag{14}$$

For *minDInc* and *minDDec* we need time quantization. This is done uniformly in discrete Δt-wide intervals, with the resulting values stored in M-bit words. Δt mustn't be larger than the minimum *minDInc/Dec*, represented by Δt_{max}, which depends on N, as seen in Eq. 5. But it can be smaller, improving time resolution, although for a fixed M, reducing the range of time quantization. For each pair of events, an M-bit comparison is needed to find the extrema:

$$Cost_{minDInc+Dec} = (\overline{\#events} - 1) \times M \tag{15}$$

TM does not require computations for feature extraction, as these are the samples themselves. However, a larger number of features are produced, increasing the memory of the system plus the computational cost of the classifier, which is:

$$Cost_{classifier} = \#classes \times (\#features \times (N + 2) - 1) \tag{16}$$

4.3 Results

In Table 3 we can see the average number of events and inversion points per spike. For the EDF, by varying N and looking at the average accuracy over all datasets we find that $M = 4$ and $\Delta t = \Delta t_{max}/4$, provides the best results, while allowing to minimize time quantization.

Table 3: Average number of events and inversion points per spike.

	N				
	3	4	5	6	7
# events	4.3	9.9	21.3	44.2	90.0
# inv. points	1.2	1.5	1.7	2.0	2.2

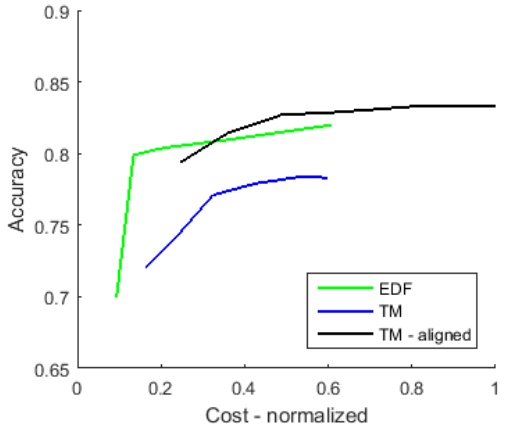

Figure 3: Average sorting accuracy over all the 16 datasets versus computational cost of spike sorting using the EDF, TM and TM after alignment.

These values are used in Figure 3, where I compare the average accuracy over all datasets versus computational cost of the proposed EDF and TM. For the EDF, N varies from 3 to 7, while for TM, N varies from 3 to 8 and the SR from 10 kHz to 20 kHz.

We can see how, for the same computational cost the proposed features outperform TM, with the lower costs of the EDF enabled by asynchronous delta-modulation. However, the cost of time quantization is not included in the reported values. It should also be noted that prior to filtering the spikes are all normalized to 1, which reduces the sorting contribution of *max*.

In Figure 3 a curve is also shown for TM after aligning spikes to their maximum, with a 0.5 ms window before the maximum and a 1 ms window afterwards. With this, performances become slightly better than the EDF, however, the reported costs do not include the additional burden of searching for the maximum and aligning spikes, which will also require using a pre-detection buffer to store the previous X samples at any moment.

It was also found that the EDF and TM with alignment display similar behaviours when changing the difficulty and noise levels of the datasets, with neither proving to be more robust in worse conditions.

5. FPGA Implementation

The proposed system was implemented on a FPGA to demonstrate how the CT outputs of a LC ADC can be used to drive asynchronous spike sorting.

A TerasIC DE0 board with Altera's Cyclone III FPGA was used. ModelSim was used to simulate the designs and Altera's Quartus II software to compile the Verilog code and program the FPGA. To output the sorting results, Saleae's logic analyser was used.

An overview of the system can be seen in Figure 4. The neural datasets were initially filtered on *Matlab*, after which the LC ADC model was applied. Time was quantized into discrete 1 μs steps, each containing 2 bits representing the REQ and DIR pulses. The step following a level-crossing must necessarily have REQ = 0, as the positive edges of REQ are used to trigger the circuitry.

The resulting binary array is preceded by setup variables for the system: the initial value of the data, to set the counter, and the detection threshold.

The data was stored on a flash memory included on the DE0 board. A memory controller module was developed to read it, storing the setup variables and streaming the REQ and DIR pulses into the system.

The detector includes a counter, incremented or decremented by 1 at each REQ pulse if the corresponding DIR is '1' or '0'. The value of the counter is then compared with the detection threshold. If larger, a window signal switches to '1', serving as an enable bit for the following feature extraction modules.

The memory controller and detection modules are the only that require a clock signal. For the latter it is used solely to set the time during which the window signal is '1' after each detection.

To manage asynchronous feature extraction and classification, delay units were used to create control pulses. These however, are not an integral part of FPGAs, as they are intended to implement synchronous circuits.

Delays were obtained by forcing a signal trough several logic cells of the FPGA. By varying their number, the length of the delay can be controlled. It should be noted that this is not a good practice, as delays obtained from the inherent propagation time of the technology will vary with temperature, power supply voltage and device fabrication, which can render the circuit unreliable. Additionally, an excessive amount of logic cells are needed to create the delays. For our demonstrative purpose, however, it suffices.

More efficient methods could be used if imple-

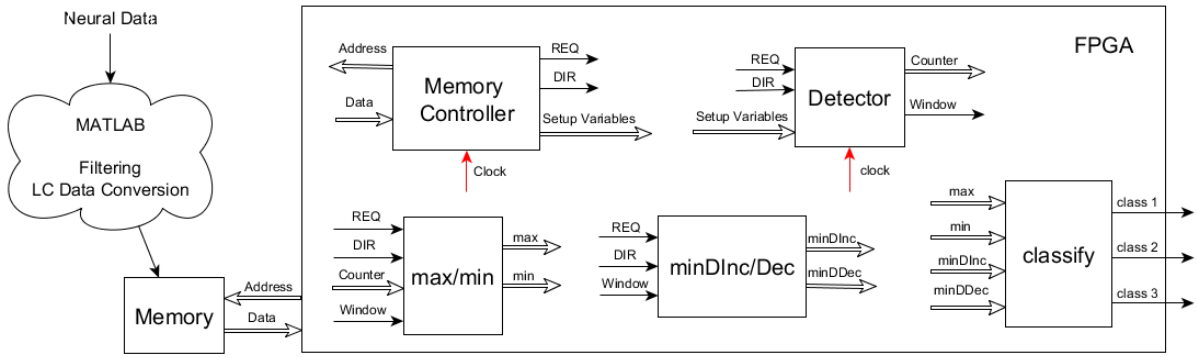

Figure 4: Overview of the FPGA implemented system. Note how the clock is only connected to the memory and detector modules. White arrows represent strings of bits.

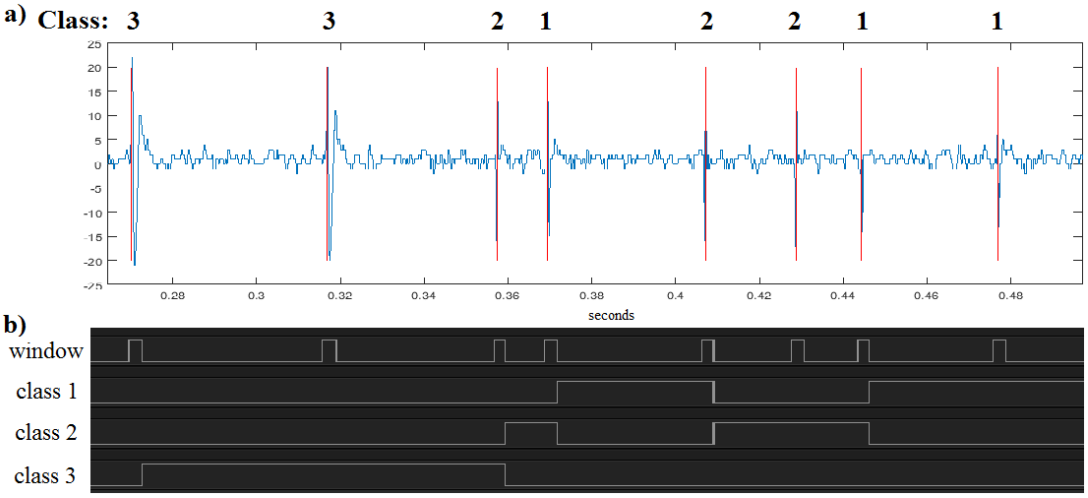

Figure 5: a) Input data and corresponding classes of each spike. b) Sorting results from the logic analyser.

menting the system on an ASIC, for example voltage controlled digital delay cells.

To quantify time, for the derivative features, an oscillator and counter are used to count the number of cycles that occur between level-crossings. The oscillator was implemented using a delay element that feeds back into its input, creating a loop. An initial pulse is created from the positive edge of the window signal, and when window switches back to '0', the oscillator is stopped, as time only needs to be quantified to extract features.

When the window signal switches to '0', the falling edge triggers classification. As opposed to the l1-norm classifier explained in Section 4, classification was achieved by comparing the features with pre-determined values, found offline. The final output are 3 bits which are individually '1' if the current spike was classified to the corresponding class. The falling edge of window is also used, after a delay, to reset the feature values for the following detection.

Results

For demonstrative purposes, a data segment from dataset 'Easy1' with noise level 0.05, was input in the system, Figure 5 a). N was set to 6 for the LC ADC, and $M = 6$ and $\Delta t = 1\ \mu s$. The length of window was 2.4 ms. The sorting results are seen in Figure 5 b).

All spikes were correctly classified, as was expected, since a 1-to-1 correspondence between *Matlab* simulation and FPGA feature values was obtained. The proposed system has thus been validated, managing to asynchronously sort spikes driven by the outputs of LC ADCs.

6. Conclusions

With this study we have confirmed LC sampling produces fewer, more informative samples than US, due to the spurious behaviour of extracellular recordings. The smallest amplitude resolution allowed to verify this condition also did not have a deleterious effect of spike sorting performances.

A continuation of this work should consider using adaptive resolution LC ADCs. Not only fewer

samples are expected, but the input dynamic range would increase and the speed requirements would relax. Adaptive resolution would have little effect on the cost of extracting the proposed EDF. *max* and *min* would be unaltered. For *minDInc* and *minDDec*, if the coarser resolutions are set as powers of 2 of the finer one, the time between level-crossings needs only to be shifted by a corresponding number of bits to produce the derivative features.

Any improvement to the design of LC ADCs in general would also be welcomed for this approach.

The proposed EDF are shown to result in similar performances to the conventional spike sorting method of TM, while being slightly computationally lighter. Their asynchronous hardware implementation is also shown to be feasible.

A future work, however, should include a more detailed cost analysis, as only computational costs were considered here. Different methods of time quantization, and corresponding implementation costs, should be explored. Also, an integrated view of the whole system should be done, quantifying the possible power savings from using an asynchronous DSP on spurious data, due to its activity dependent dynamic power consumption, and from requiring less memory by processing the inputs as they arrive. This will require changing the cost function from numbers of operations to power consumption and silicon area, possibly requiring deeper consideration for the technology used.

References

[1] S. Todorova et al. To Sort or Not to Sort: The Impact of Spike-Sorting on Neural Decoding Performance. *Journal of Neural Engineering*, 11(5), Oct 2014.

[2] C. Gold et al. On the Origin of the Extracellular Action Potential Waveform: A Modelling Study. *Journal of Neurophysiology*, 95:3113–3128, Feb 2006.

[3] I. H. Stevenson et al. How Advances in Neural Recording Affect Data Analysis. *Nature Neuroscience*, 14:139–142, Feb 2011.

[4] S. Kim et al. Thermal Impact of an Active 3-D Microelectrode Array Implanted in the Brain. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(4):493–501, Dec 2007.

[5] Deren Y. Barsakcioglu. *Resource Efficient On-Node Spike Sorting*. PhD thesis, Imperial College London, December 2015.

[6] R. Xu et al. Clustering Algorithms in Biomedical Research: A Review. *IEEE Reviews in Biomedical Engineering*, 3:120–154, Oct 2010.

[7] I. Williams et al. A Scalable 32 Channel Neural Recording and Real-Time FPGA Based Spike Sorting System. *IEEE Biomedical Circuits and Systems Conference*, pages 187–191, Oct 2015. Atlanta, USA.

[8] D. Y. Barsakcioglu et al. A 32-Channel MCU-based Feature Extraction and Classification for Scalable On-node Spike Sorting. *IEEE ISCAS*, pages 1310–1313, May 2016.

[9] A. Bonfanti et al. A Low-Power Integrated Circuit for Analog Spike Detection and Sorting in Neural Prosthesis Systems. *IEEE Biomedical Circuits and Systems Conference*, pages 257–260, Nov 2008.

[10] V. Karkare et al. A 130-μW, 64-Channel Neural Spike-Sorting DSP Chip. *IEEE Journal of Solid-State Circuits*, 46:1214–1222, May 2011.

[11] V. Karkare et al. A 75-μW, 16-Channel Neural Spike-Sorting Processor With Unsupervised Clustering. *IEEE Journal of Solid-State Circuits*, 48(9):2230–2238, Sep 2013.

[12] U. Rutishauser et al. Online Detection and Sorting of Extracellularly Recorded Action Potentials in Human Medial Temporal Lobe Recordings, in vivo. *Journal of Neuroscience Methods*, 154:204–224, 2006.

[13] H. Inose et al. Asynchronous Delta-Modulation System. *Electronics Letters*, 3:95–96, 1966.

[14] X. Zhang et al. A 300-mV 220-nW Event-Driven ADC With Real-Time QRS Detection for Wearable ECG Sensors. *IEEE Transactions on Biomedical Circuits and Systems*, 8(6):834–843, Dec 2014.

[15] Z. Zhu et al. A 0.6-V 38-nW 9.4-ENOB 20-kS/s SAR ADC in 0.18-μm CMOS for Medical Implant Devices. *IEEE Transactions on Circuits and Systems*, 62(9):2167–2176, Sep 2015.

[16] R. Q. Quiroga et al. Unsupervised Spike Detection and Clustering with Wavelets and Superparamagnetic Clustering. *Neural Computation*, 16(8):1661–1687, Aug 2004.

[17] Y. Li et al. A Continuous-Time Programmable Digital FIR Filter. *IEEE Journal of Solid-State Circuits*, 41(11):2512–2520, Nov 2006.