Algorithms and Architectures for Low Power Spike Sorting

Alex Zviagintsev, Yevgeny Perelman and Ran Ginosar

VLSI Systems Research Center,

Technion—Israel Institute of Technology, Haifa 3200, Israel

[alehan@tx.technion.ac.il]

Abstract: Front-end integrated circuits for signal processing are useful in neuronal recording systems that engage a large number of electrodes. Detection, alignment, and sorting of the spike data at the front-end reduces the data bandwidth and enables wireless communication. Without such data reduction, large data volumes need to be transferred to a host computer and typically heavy cables are required which constrain the patient or test animal. We explore Neuroprocessor electronic chips for portable applications. The Neuroprocessor can be placed next to the recording electrodes and provide for all stages of spike processing, stimulating neuronal tissues and wireless communication to a host computer. It can dissipate only a limited amount of power, due to supply constraints and heat restrictions. We introduce hardware architectures for automatic spike sorting algorithms in Neuroprocessors, designed for low power. Some of the algorithms are based on principal component analysis. Others employ a novel Integral Transform analysis and achieve 98% of the precision of a PCA sorter, while requiring only 2.5% of the computational complexity. The algorithms execute autonomously, but require off-line training and setting of computational parameters. We employ pre-recorded neuronal signals to evaluate the accuracy of the proposed algorithms and architectures: The recorded data are processed by a standard PCA spike sorting software algorithm, as well as by the several hardware algorithms, and the outcomes are compared.

I. INTRODUCTION

Electrophysiological study of brain structures using wire electrodes is one of the oldest methods in neuroscience [1]. A single electrode can often pick up signals of multiple neurons from a small region around its tip. Sorting action potential waveforms ("spikes") originating from different neurons can be performed either on-line [2]-[5] or off-line [6]-[8] using various methods for pattern recognition. On-line sorting is required for closed loop experiments (in which stimulations are generated in response to detected spikes [9]) and for clinical applications [10]. Off-line sorting is sometimes employed to compute the parameters required for on-line algorithms.

On-line sorting requires high bandwidth communications between the electrodes and the sorting computer, as all the recorded data has to be transferred to the host. When a large number of signals is to be handled, typical computing resources are insufficient [11]. Special-purpose hardware for spike processing is called for high-volume research and clinical applications.

Implantable integrated circuits with front-end processing of spikes can significantly reduce the communication bandwidth with the back-end computer by transferring only the outcome of the sorting algorithm. For instance, given a sampling rate of 24Ksps and 12 bit sampling precision, the raw data rate is 288Kbits/second per electrode. Spike sorting converts each spike to a short *spike notification message* (~20 bits). Assuming a high rate of 100 spikes/sec/electrode, the required data rate is reduced down to 2Kbits/sec per electrode, less than 1% of the original rate.

We investigate the Neuroprocessor, a dedicated integrated circuit for front end analog processing, conversion to digital [12], and spike sorting [13]. If the Neuroprocessor is to be implanted in the brain, and/or used in portable applications (such as neuro-prosthetics), it should require minimal power for its operation. The purpose of this study is to investigate spike sorting algorithms and architectures that minimize power consumption. Such algorithms trade off some classification accuracy in return for significant savings in power.

The feasibility of hardware implementation of two spike-sorting algorithms from a power consumption point of view was discussed in [14]. Yet, typical spike-sorting algorithms, such as based on Principal Component Analysis (PCA [15]), are unattractive for efficient implementation in hardware, as they require storing and iterative processing of large amounts of data. We consider five algorithms and architectures that achieve high precision spike sorting while minimizing power dissipation, and investigate their sorting errors, relative to a standard PCA software algorithm [16][17]. All algorithms require off-line training for setting their computational parameters. Periodic re-training is typically required.

The *Hard Decision* algorithm compares the spike signal with predetermined values. The *Soft Decision* algorithm applies filtering prior to making the comparisons. The *Integral Transform* algorithm applies linear signal separation in a predetermined integral transform space. The *PCA sorting* architecture implements common PCA with linear classification, and the *Segmented PC* algorithm applies PCA with reduced precision.

Section II presents an overview of the system. The various algorithms and architectures are described in Section III, and their performance is analyzed in Section IV.

II. System Overview

In typical experimental setups, the signals recorded by the electrodes are amplified and transmitted over wires to a host computer where they are digitized and processed according to the experimental requirements [18]. The main disadvantage of that experimental arrangement is the need to connect a cable to the subject, restricting its movement. The Neuroprocessor performs front-end data processing and reduction, to enable bidirectional wireless communication that replaces cables and allows free movement of the patient or test subject.

A wireless neuronal recording system comprises a Neuroprocessor, electrodes, a wireless modem, and a power source. While it is feasible to transfer some raw signal recordings over the wireless channel, the large volume of data collected by many electrodes is prohibitive [11], and data reduction must be carried out by the Neuroprocessor. In many neuronal experiments, the most important data is the indication of spikes, their sources (electrode and identifiable unit within the electrode), and the time of their occurrence. These indications are produced with a real-time hardware spike sorting algorithm; the Neuroprocessor transmits only the spike indications and avoids sending the raw signal. Such indications require much lower communication bandwidth and could be made feasible with low-power wireless links. As indicated above, spike sorting may lead to more than 99% reduction in the data bandwidth requirements.

The architecture of the part of the Neuroprocessor that processes the signal from a single electrode is shown in Figure 1. The signal is first amplified and digitized. The Spike Detector detects the presence of spikes in the input, determines their starting point, and initiates the operation of the Spike Sorter. Both spike detection and sorting must be adaptable, due to unstable recording conditions. Therefore, raw data is periodically transmitted to the host computer for retraining and recalculated parameters are sent back to the Neuroprocessor. At all other times, the spike signal is processed by the Spike Detector and Spike Sorter. In this paper we focus on spike sorting algorithms and assume a given Spike Detector. The output logic produces the spike notification message.

III. Hardware Algorithms and Architectures for Spike Sorting

When exploring VLSI architectures for real time spike sorting to be carried out at the head-stage, we seek to minimize the required resources while still achieving acceptable levels of accuracy. The primary goal is to minimize power dissipation. Following [14], we consider the relative computational complexity of a few architectures as a predictor of their power measure.

Five different hardware sorting algorithms are considered. The first two algorithms, *Hard Decision* (HD) and *Soft Decision* (SD), perform classification in the time domain. The remaining three algorithms, *Integral Transform* (IT), *Principal Component Sorting* (PC) and *Segmented PC* (k-PC), classify in transform domains.

A. The Hard Decision (HD) Algorithm

The *Hard decision* (HD) algorithm compares the spike signal with a pre-computed separation line. It is relatively simple to implement in VLSI and incurs a low computational complexity, potentially requiring small circuit area and dissipating low power. However (as discussed in Section IV below), it is sensitive to noise, resulting in rather high classification errors.

The HD algorithm operates as follows: The preliminary recorded spikes in the training set are clustered into separate groups. Clustering can be human supervised [3][19] or unsupervised [20]-[22]. Figure 2a shows an example of two clusters in a space spanned by the first two principal components ("PC space") [15]. The separation line is represented by the black waveform. Sometimes it can be generated by the inverse PCA transform of a "center point" (on the PC space) that is placed between the two clusters (Figure 2b). However, when the clusters are too close to each other, the separation line may need to be generated directly on the time series representation (Figure 2a). The spike signal can then be split into several time intervals, according to the positive and negative phases of the spikes (marked A, B in Figure 2a). For each time interval we can determine whether a spike of a specific cluster is expected to have values either above or below the separation line. For instance, spikes of the cluster colored green (light) are above the separation line in time interval A and below it in time interval B. This algorithm may also be generalized for cases where more than two separate units (neurons) are identifiable.

A VLSI architecture for the HD algorithm is shown in Figure 3. The input x(i) is compared to the synchronized value of the separation line M(i). The comparator outputs are series of bits, which are accumulated into SA and SB, during the A and B time intervals, respectively. These two sums represent the number of signal points within the respective time intervals that are above the separation line. In the second stage, these sums are compared with the sorting threshold values TA and TB. If the two comparisons agree with the predefined values, a "spike notification message" is sent to the host computer. For instance, a spike from the green (light) cluster (Figure 2a) would generate a high number in time interval A (ideally equal to the number of samples included in A) and a low number (ideally zero) in time interval B. The computational complexity and sorting performance (in terms of errors relative to an off-line PCA algorithm) are discussed in Section IV below.

Formally, consider a discrete-time signal input:

$$\left\{ x(i) \mid i = 1, \ldots, N \right\},$$

two index sets (corresponding to two decision intervals in the time domain):

$$A = \left\{ x(i) \mid i_1 \le i \le i_2 \right\}, \; B = \left\{ x(i) \mid i_3 \le i \le i_4 \right\},$$

and a given discrete-time separation line $\left\{ M(i) \mid i = 1, \ldots, N \right\}$. Note that all given factors (short of the input signal) are pre-computed by means of off-line training. First, compute two sums:

$$S_A = \sum_{i \in A} \left(x(i) > M(i) \right),$$
$$S_B = \sum_{i \in B} \left(x(i) > M(i) \right),$$

Then the classification is made as follows:

$$Class = \begin{cases} I, & if\ \ U_A S_A - T_A \geq 0\ \ and\ \ U_B S_B - T_B \geq 0 \\ II, & if\ \ U_A S_A - T_A < 0\ \ and\ \ U_B S_B - T_B < 0 \\ unsorted, & otherwise. \end{cases} \qquad (1)$$

where T_A, T_B are two threshold values and $U_A, U_B \in \{-1, 1\}$ are two sign values that indicate whether S is expected to be larger or smaller than T for each of the two time intervals and for each of the two clusters. The sorting performance of the HD algorithm may be improved by considering more than two decision intervals.

B. The Soft Decision (SD) Algorithm

The *Soft Decision* (SD) algorithm is similar to the HD algorithm, with classification made as follows: the input signal is integrated for each of the pre-defined time intervals and the integrals are compared with the respective integrals of the separation line. Note that while the HD algorithm sums up (within each time interval) the results of comparing the signal with the separation line, the SD algorithm on the other hand performs the summation first and then compares the signal with the separation line, potentially resulting in improved classification accuracy thanks to lower noise sensitivity.

A VLSI architecture for the SD algorithm is shown in Figure 4. The input is integrated within each time interval (A, B). The two integrals S_A and S_B are compared with the predefined threshold values T_A and T_B. If the two integrals fall within the expected ranges, the Neuroprocessor issues a “spike notification message” to the host computer.

The SD algorithm may be perceived as operating on an Integral Transform (IT) space instead of the PC space (See Figure 5b). The two axes represent the normalized values of the signal integrals over A and B. The integrals of the separation line result in a single point in IT space. Similar to the HD classification, the time-domain comparisons made by the SD algorithm effectively define two recognition quadrants in IT space on opposite sides of the “center point.” Such a separation criterion does not obtain reliable results for closely located clusters—many points from both clusters may fall into "unrecognized" quadrants and as a result are unclassified (See Section IV below, Figure 11).

The formal definition of SD is similar to HD, except for the definition of S_A, S_B, T_A and T_B:

$$\{S_A, S_B\} = \left\{ \frac{1}{N_A} \sum_{i \in A} x(i), \quad \frac{1}{N_B} \sum_{i \in B} x(i) \right\}$$

$$\{T_A, T_B\} = \left\{ \frac{1}{N_A} \sum_{i \in A} M(i), \quad \frac{1}{N_B} \sum_{i \in B} M(i) \right\}$$

where $N_A = i_2 - i_1 + 1$, $N_B = i_4 - i_3 + 1$, and $\{M(i) \mid i = 1,...,N\}$ is a given separation line. Classification is performed according to Eq. (1).

C. The Integral Transform (IT) Algorithm

The *Integral Transform* (IT) sorting algorithm classifies the spikes projected in the two-dimensional Integral Transform space. The two axes of the IT space represent the normalized signal integrals over the time intervals A and B (Figure 5b):

$$I_A = \frac{1}{N_A}\sum_{i=1}^{N_A} x(i), \quad N_A = i_2 - i_1 + 1$$
$$I_B = \frac{1}{N_B}\sum_{i=1}^{N_B} x(i), \quad N_B = i_4 - i_3 + 1$$

Where N_A and N_B are the number of samples in A and B, respectively.

Both IT and SD algorithms apply a similar integration step, but whereas the SD algorithm employs time domain discrimination, the IT algorithm uses linear classification in IT space, as follows:

$$Class = \begin{cases} I, & if\ I_B > m \cdot I_A + n \\ II, & if\ I_B < m \cdot I_A + n \end{cases}$$

Here *m* and *n* are the parameters of the separation line (Figure 5b). They are determined by off-line learning, which may be based on any appropriate technique, such as SVM [23].

Linear classification has been selected in an attempt to minimize hardware and computational complexities. In simple cases, one line may suffice for sorting spikes into two clusters. In general, any number of lines may be employed, either to further constrain the classification space, or to enable sorting into three or more clusters, or both. An example of PC and IT classification into three clusters by two lines is shown in Figure 6. Comparing the signal against each line requires one multiplication, one addition and one comparison.

The transformations of a given signal into the PC and IT spaces, as seen in Figure 5a and b, appear similar. Both transformations apply linear FIR filters, whose step response is either the principal components (for PC) or rectangular (for IT). The integration intervals are ideally positioned at times when signals from different clusters are expected to differ the most, providing for a good discrimination in IT space. In Section IV below we show empirically that the IT algorithm can achieve classification to within 2.2% of PCA.

A VLSI architecture for the IT algorithm is shown in Figure 7. The input is integrated over the first time interval (A) and the result is stored as I_A (storage is omitted from the figure). During the second interval (B), the integrator generates I_B. Subsequently, for each possible dividing line, the parameters m and n are used to generate the corresponding mI_A+n value that is compared with I_B.

D. Principal Component Sorting

A VLSI architecture for on-chip sorting by means of principal component analysis (PCA) using two principal components and linear classification is shown in Figure 8. Each input sample is multiplied by two PC coefficients, and the two accumulated projections are linearly compared, similarly to the IT algorithm.

E. Segmented PC Sorting

The segmented PC ("K-PC") algorithm approximates PCA by down sampling the principal component vectors, reducing the number of multiplications. The signal is integrated over several time intervals (k_1 intervals for PC_1 and k_2 intervals for PC_2). Each integral is multiplied by the average of the principal component values over the same interval, as follows:

$$S_1 = \sum_{p=1}^{k_1} I_1(p) \cdot \alpha(p)$$

where

$$I_1(p) = \sum_{i \in \text{interval } p} x(i)$$

and

$$\alpha(p) = \underset{i \in \text{interval } p}{Average}\left\{ PC_1(i) \right\} \quad .$$

The expressions for PC_2 are similar. A VLSI architecture for the K-PC algorithm is shown in Figure 9.

Another level of savings in computational complexity can be achieved by approximating the multiplication coefficients K-$PC_{1,2}$ by powers of 2 (achieved by simple bit-shifting).

IV. RESULTS

A. Algorithm Validation

The hardware spike sorting algorithms described above are compared (by simulation) to a software implementation of PCA [16]. Details of the PCA sorting are described in [17]. All algorithms are applied to the same data set, comprising a large number of digitized recorded spikes. Figure 10 illustrates the algorithm validation scheme. First, a part of the data is used for training, producing configuration parameters for the hardware algorithm. Second, the parameters are downloaded to the Neuroprocessor. Third, a simulation of the Neuroprocessor hardware spike-sorting algorithm is applied to the entire data set. After the software (PCA) algorithm is also executed on the same data, the results are compared.

B. Spike Recording Method

Spike data was taken from electrophysiological recordings of multiple spike trains, obtained from microelectrodes implanted in various cortical regions [24]. Neuronal signals from the electrodes were amplified, bandpass filtered (300 – 6000 Hz, four poles Butterworth filter), and sampled at 24 Ksps/electrode. Spike detection was done offline [17][25]; only stable spike trains (as judged by stable spike waveforms, stable firing rate and consistent responses to behavioral events) were included in this study. The data set contains about 1000 spikes per cluster.

The hardware algorithms have been applied to this data under a number of simplifying assumptions, ignoring classification errors such as overlapping signals, burst-firing neurons and non-stationary background noise. The study focuses on the basic problem of hardware-based classification of single neuron spikes contaminated by noise.

C. Analysis of Spike Sorting Algorithms

The reduced computational complexity of the proposed VLSI algorithms comes at the expense of precision. Two types of errors emerge when the VLSI algorithms are compared with software PCA sorting: A spike may be *unclassified* or *mis-classified* by a hardware algorithm. The two error types are combined into a cumulative error rate, which serves as a figure of merit for the algorithm.

Another measure of the various architectures relates to their computational complexity, which is roughly related to the power consumption. We count the number of additions and multiplications required for every algorithm to process a single spike. Multiplication is counted as about ten additions. Computational complexity is expressed in the total number of equivalent additions.

1) HD and SD Algorithms

Classification errors of the HD and SD algorithms range from 10% to 25%, for clusters that are closely located in the PC space (see Table 1). The vast majority of errors is due to unclassified spikes. Depending on the goals of the experiment, these error rates may or may not be acceptable.

The low performance of the HD and SD algorithms relates to the fact that they perform time-domain classification. This type of crude classification can be represented as two recognition quadrants on the opposite sides of the "center point" in IT space (Figure 11). This separation criterion does not obtain reliable results, especially for closely located clusters: Many spikes from both clusters fall into the adjacent quadrants and as a result are unclassified.

2) The IT Algorithm

The IT algorithm was developed to address the high sorting errors of the HD and SD algorithms. The IT algorithm separates the clusters by means of lines (Figure 5b), thus eliminating the unclassified regions. The IT algorithm reduces the error rate to 2.2% (Table 1). It is evident in Figure 12b that the IT algorithm incurs the least computational complexity among all low-error, low-complexity solutions.

3) The Segmented PC Algorithm

Four versions of the segmented PC algorithm have been studied: seven and fifteen segments were simulated both with full and reduced precision. In reduced precision we have taken coefficients of the form 2^n, reducing the cost of multiplication to a single addition.. The segmented PC algorithms achieve error rates that are somewhat better than IT, at the cost of increased complexity. Note that their computational complexity is more than an order of magnitude better than a full PCA.

Computational complexity and error rates are compared in Table 1and Figure 12. Three conclusions are proposed: First, time-domain classification results in high error rates and does not yield a complexity advantage relative to IT. Second, IT classification constitutes the "knee point" of the complexity versus error graph, and is thus suggested as the preferred architecture. Third, PCA-based classification algorithms incur an order of magnitude lower complexity than the full-fledged PCA, yet, they yield only a marginal error performance over IT.

V. CONCLUSIONS

We have considered low-power spike sorting algorithms and suitable architectures. Such systems may be useful for implanting near the recording electrodes, or for using in large multi-electrode arrays, in either research or clinical applications. They enable substantial reduction of the communication bandwidth, which is essential when a large number of recording electrodes is involved.

We have described five VLSI algorithms for spike sorting: *Hard Decision* (HD) compares the signal with pre-computed thresholds. *Soft Decision* (SD) integrates the signal in segments and compares the integrals with pre-computed thresholds. *Integral Transform* (IT) also integrates the signal in segments but uses the results for linear classification in a two-dimensional "integral space." PC implements a common PCA analysis with linear classification, and segmented PC applies PCA in segments.

The algorithms have been simulated with real neuronal spike data. The results are analyzed in terms of classification errors (relative to sorting achieved with software PCA classification). The computational complexity of each algorithm was estimated based on the number of additions and multiplications involved.

The two time-domain classification algorithms, HD and SD, produce high error rates. The segmented PC algorithms incur significantly lower complexity than full PCA. They yield only a marginal error performance over IT, while requiring higher complexity. The IT algorithm turns out to constitute the "knee point" of the complexity versus error graph, and is thus suggested as the preferred architecture for implementation in Neuroprocessors.

Acknowledgment

This research was funded in part by a grant from the Office of Chief Scientist, Israel Ministry of Industry and Trade. Guidance from Moshe Abeles, Hagai Bergman, Eilon Vaadia, Shimon Marom, Izhar Bar-Gad and Alpha-Omega, Inc. is greatly appreciated.

References

[1] Gesteland R.C., Howland B., Lettvin J.Y., and Pitts W.H., "Comments on microelectrodes," *Proc. IRE*, vol. 47, pp. 1856–1862, 1959.

[2] Worgotter F., Daunicht W.J. and Eckmiller R., "An on-line spike form discriminator for extracellular recordings based on an analog correlation technique," *J. Neurosci. Methods*, 17:141-151, 1986.

[3] Chandra R. and Optican L.M., "Detection, classification, and superposition resolution of action potentials in multiunit single channel recordings by an on-line real-time neural network," *IEEE Trans. Biomed. Eng.*, 44: 403–412, May 1997.

[4] Fee M.S., Mitra P.P. and Kleinfeld D., "Automatic sorting of multiple-unit neuronal signals in the presence of anisotropic and non-gaussian variability," *J. Neurosci. Methods,* 69: 175–188, 1996.

[5] D'Hollander E. H. and Orban G., "A Spike recognition and on-line classification by an unsupervised learning system," *IEEE Trans. Biomed. Eng.* 26: 279–84,1979.

[6] Jansen R.F., "The reconstruction of individual spike trains from extracellular multineuron recording using a neural network emulation program," *J. Neurosci. Methods*, 35: 203-213, 1990.

[7] Plexon Inc. http://www.plexoninc.com/ofs.htm. Dallas, TX, 2003.

[8] Lewicki, M.S., "A review of methods for spike sorting: the detection and classification of neural action potentials." *Network: Comp. Neural Syst.* 9(4): 53-78, 1998.

[9] Shahaf G. and Marom S., "Learning in networks of cortical neurons," *J. Neurosci. Methods*, 21(22): 8782-8788, 2001.

[10] Mussa-Ivaldi F.A. and Miller L.E., "Brain–machine interfaces: computational demands and clinical needs meet basic neuroscience," *Trends Neurosc.,* 26: 6, 329-334, 2003.

[11] Harrison R., "A low-power integrated circuit for adaptive detection of action potentials in noisy signals," *Proc. 2003 Intl.Conference of the IEEE EMBS*, Sept. 17-21, 2003.

[12] Perelman Y. and Ginosar R., "An Integrated System for Multichannel Neuronal Recording with Spike / LFP Separation and Digital Output," *2nd Int. IEEE EMBS Conf. Neural Eng.,* 377-380, 2005.

[13] Zviagintsev A., Perelman Y. and Ginosar R., "Low-Power Architectures for Spike Sorting," *2nd Int. IEEE EMBS Conf. Neural Eng.,* 162-165, 2005..

[14] Zumsteg Z.S., Ahmed R.E., Santhanam G., Shenoy K.V., Meng T.H., "Power Feasibility of Implantable Digital Spike-Sorting Circuits for Neural Prosthetic Systems," *Proc. 26th Ann. Int. Conf. IEEE EMBS*, pp. 4237-4240, Sept. 2004.

[15] Abeles M, Goldstein M.J., "Multispike train analysis," *IEEE Trans. Biomed. Eng*., 65:762–73, 1977.

[16] Alpha-Omega Engineering Co., Ltd, Alpha Sort Ref. Manual, 1996-2002.

[17] Bar-Gad I., Ritov Y., Vaadia E., and Bergman H., "Failure in identification of multiple neuron activity causes artificial correlations," *J. Neurosci. Methods*., 107: 1-13, 2001.

[18] Nicolelis M.A.L., "Actions from thoughts," *Nature*, 409:403-407, 2001.

[19] Kim K.H. and Kim S.J., "Neural spike sorting under nearly 0 dB signal-to-noise ratio using nonlinear energy operator and artificial neural network classifier," *IEEE Trans. Biomed. Eng.* 47: 1406–1411, 2000.

[20] Lu S.N., Ekstrom A., Isham E., Fried I., Steinmetz P.N., "A Comparison of Unsupervised Automatic and Manual Cluster Analysis for Classifying Human Single Unit Data," *Society for Neuroscience Meeting*, 2003.

[21] Sahani M.M., "Variable models for neural data analysis," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, 1999.

[22] Zouridakis G. and Tam D.C., "Identification of reliable spike templates in multi-unit extracellular recordings using fuzzy clustering," *Comp. Meth. Prog. Biomed*. 61: 91–98, 2000.

[23] Vapnik V.N.,*" Statistical learning Theory"*, NY John Wiley & Sons, 1998.

[24] Eytan D., Brenner N. and Marom S., "Selective Adaptation in Networks of Cortical Neurons, " *J. Neurosci. Methods*. 23(28):9349-9356, 2003.

[25] Alpha Omega Engineering Ltd., Multi-Spike Detector (MSD).

VI. List of Figures and Tables

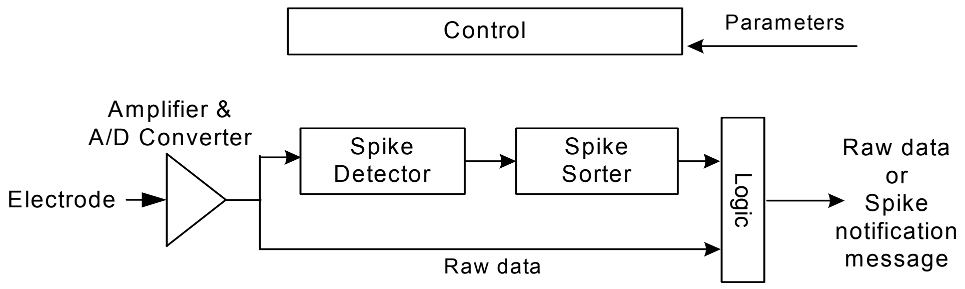


Figure 1: Neuroprocessor Architecture: one electrode section.

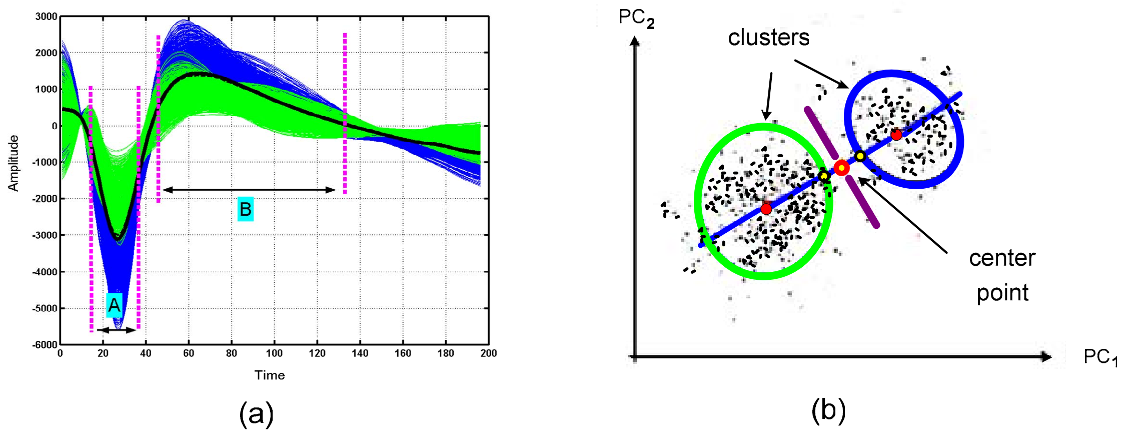


Figure 2: (a) Inversely transformed signals, (b) PC space representation.

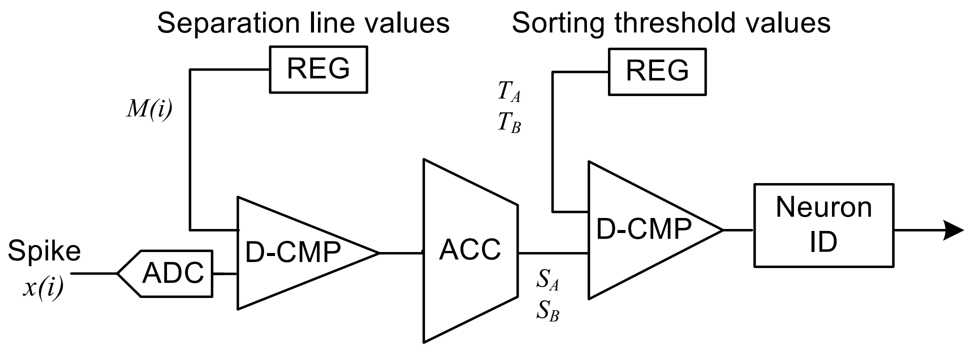


Figure 3: HD – VLSI architectures

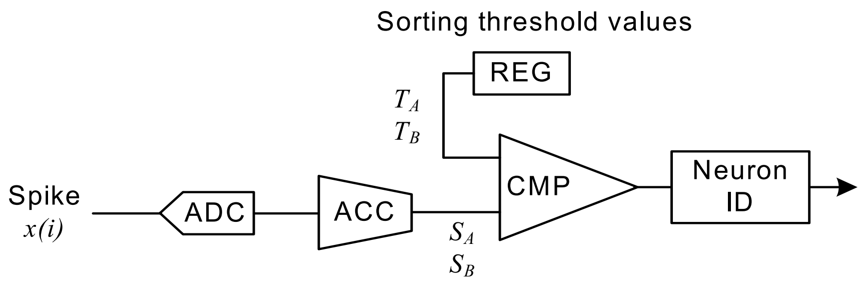
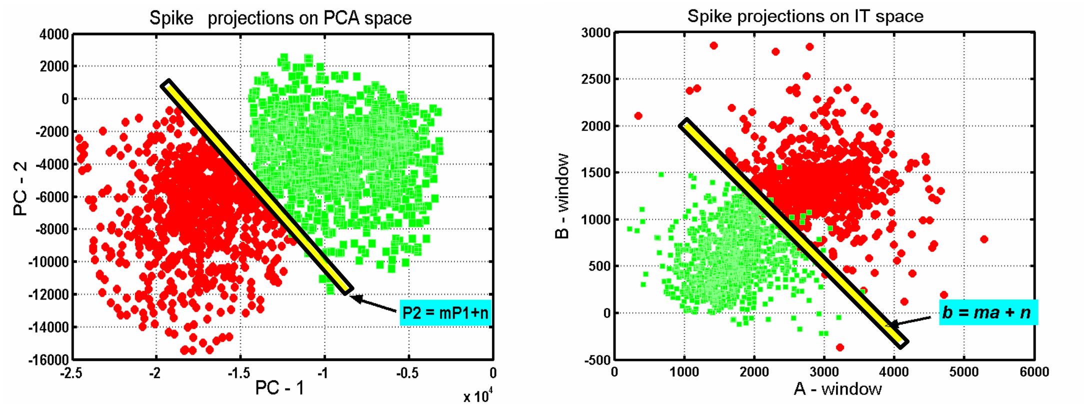


Figure 4: SD – VLSI architecture.



(a) (b)

Figure 5: Clustered signals on the Principal Component (PC) space (a) and on the Integral Transform (IT) space (b).

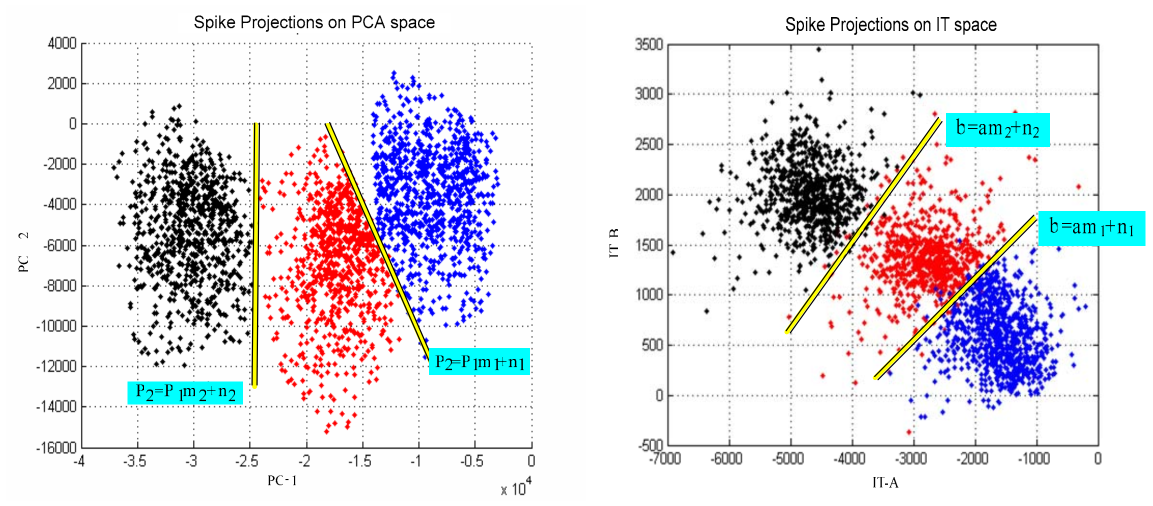


Figure 6: PC and IT classification of three clusters

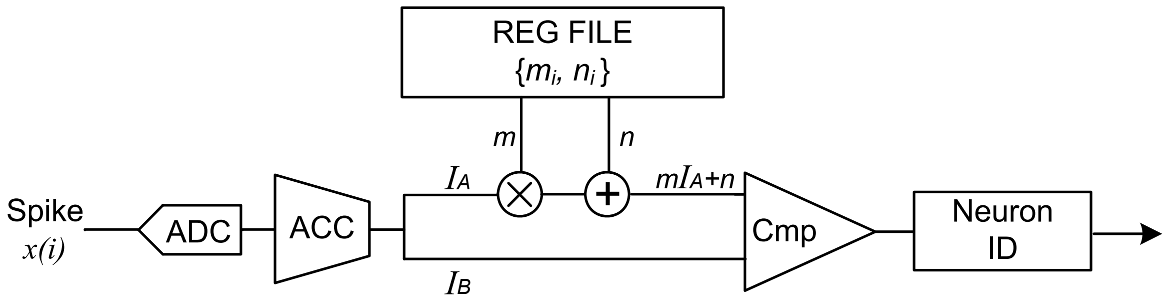


Figure 7: IT – VLSI architecture

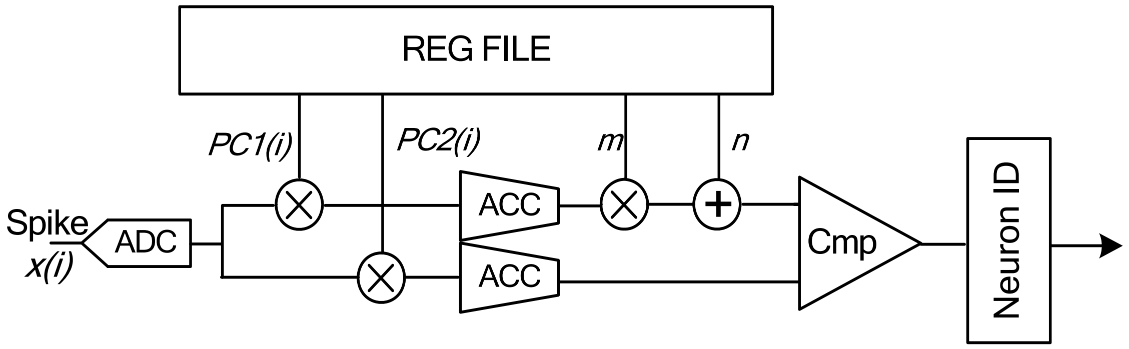


Figure 8: PC – VLSI architecture

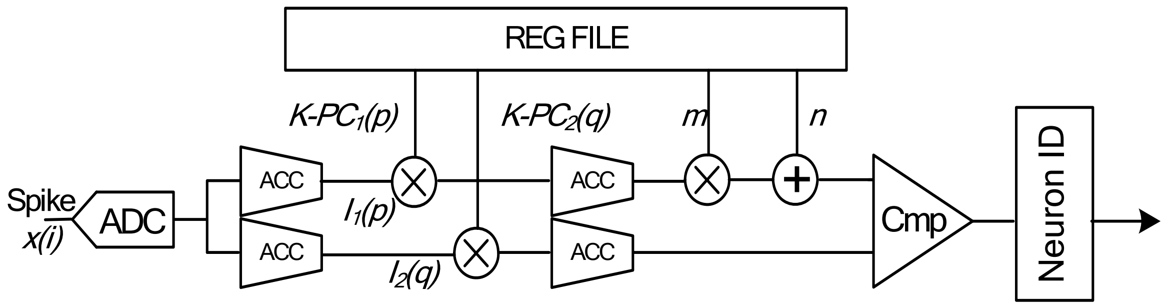


Figure 9: Segmented PC – VLSI architecture

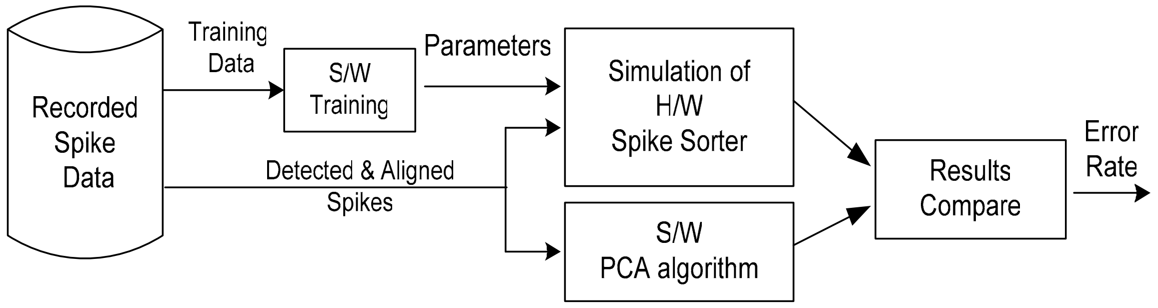


Figure 10: Algorithm validation

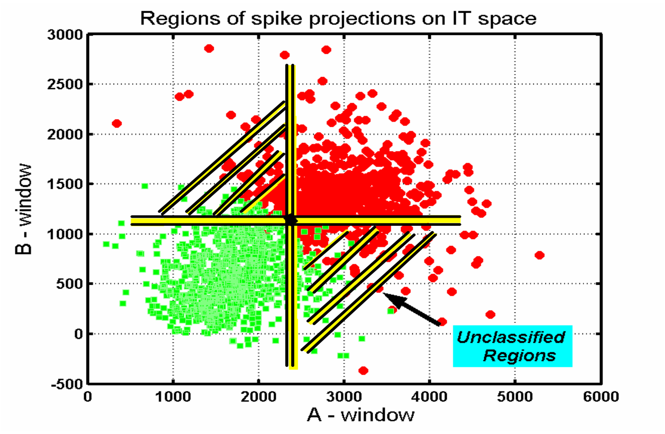
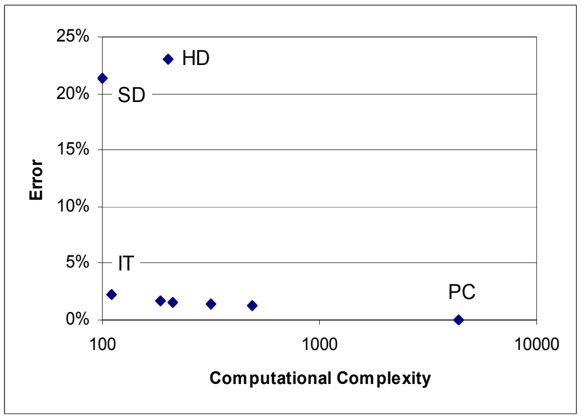
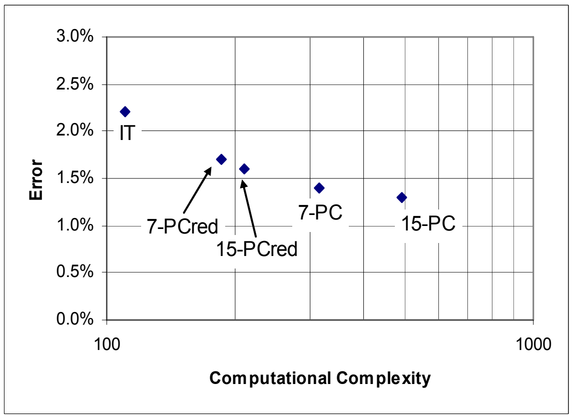


Figure 11: Classification regions for HD and SD algorithms



(a)



(b)

Figure 12: (a) Classification error vs. computational complexity,
(b) Close-up on low-complexity, low-error architectures

Algorithm	Add	Mult	Comput Compelxity	Unclass	Misclass	Error Rate
HD	200		200	20%	2.7%	23%
SD	100		100	19%	2.2%	21%
IT	100	1	110	0.8%	1.4%	2.2%
7-PC	165	15	315	0.4%	1.0%	1.4%
15-PC	190	30	490	0.4%	0.9%	1.3%
7-PC red	175	1	185	0.5%	1.2%	1.7%
15-PC red	200	1	210	0.5%	1.1%	1.6%
PC	400	400	4400	0.0%	0.0%	0.0%

Table 1: Computational complexity and classification errors of the hardware spike sorting architectures