FPGA-based neural probe positioning to improve spike sorting with OSort algorithm

László Schäffer*, Zoltán Nagy†, Zoltán Kincses,* and Richárd Fiáth‡
*Faculty of Science and Informatics, University of Szeged, Szeged, H-6725
Email: {schaffer,kincsesz}@inf.u-szeged.hu
†Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, Budapest, H-1083
Email: nagy.zoltan@itk.ppke.hu
‡Institute of Cognitive Neuroscience and Psychology, Hungarian Academy of Sciences, Budapest, H-1117,
Email: fiath.richard@ttk.mta.hu

***Abstract*—The extracellular measurement of brain electrical activity contains local field potentials and mixtures of action potentials generated by the neurons. It is essential to determine which individual neuron produces the recorded unit activity, so spike sorting methods are used. High channel-count neural probes are capable of recording the activity of large neural ensembles from up to more than hundred individual brain positions simultaneously, pose an even greater challenge for spike sorting applied on general-purpose hardware. Real-time clinical applications could greatly benefit from a hardware-accelerated data processing, especially in the case of Field-Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs), which are energy-efficient compared to traditional CPUs or GPUs, and can significantly reduce the computation time required to process large amounts of high-dimensional data. In this paper, we present a real-time FPGA-based implementation of a multi-channel Online Sorting (OSort) algorithm to pre-cluster neural data. Based on this pre-processing the neurobiologists can fine-tune the position of neural probe and improve the efficiency of offline spike sorting.**

I. Introduction

The human brain is probably the most complex system in the universe consisting of billions of neurons and quadrillions of synaptic connections. To probe this organ and to investigate fundamental neuronal mechanisms and higher-order brain functions, such as perception, learning and memory, an experimental technique used widely by neuroscientists can be applied: the extracellular measurement of brain electrical activity [1]. Commonly, one or several neural implants with multiple electrodes are lowered into the brain tissue to record the brief, electrical impulses (called action potentials or spikes) generated by neurons. The recorded, in most cases multi-channel signal is the sum of the spikes fired by several neurons surrounding the recording sites of neural probes.

To separate the spike trains of individual neurons (single-unit activity) from the recorded multi-unit activity, usually a method called spike sorting is applied [2]. Spike sorting is used in basic neuroscience research during the offline analysis of experimental data (e.g. to study the dynamics of neural networks [3]) and also in real-time clinical applications (e.g. neuroprosthetic devices, brain-machine interfaces [4], [5]). A typical spike sorting algorithm contains several computationally intensive steps (detection, feature extraction and clustering), which makes the real-time processing of data cumbersome and can degrade the performance of clinical applications requiring rapid feedback. The spike detection is based on an amplitude threshold, which can be automatically calculated using standard deviation, median absolute deviation, energy operator, wavelet transform or probability theory. During the feature extraction process the spike shapes are differentiated based on features like spike amplitude, spike width or a region of the spike itself. However more complex methods can be used to extract features, such as Principal Component Analysis (PCA) or wavelet transformation. The last step is to determine the similarity between spikes. The similar spikes are probably generated by the same neuron and can be grouped into a cluster. Clustering algorithms are based on neural network, statistics, probability, fuzzy, superparamagnetic or distance based classification [6], [7]. Spike sorting algorithms can apply offline or online and supervised or unsupervised classification methods for clustering. The offline classification works on pre-recorded neural data, while the online algorithms make the classification during the measurement. Supervised classification algorithms are using a ground truth to teach the algorithm how spikes looks like, and which spikes are similar. On the other hand unsupervised algorithms teach themselves on the fly and no ground truth is necessary [7]. High channel-count neural probes, capable of recording the activity of large neural ensembles from up to more than hundred individual brain positions simultaneously [8], pose an even greater challenge for spike sorting applied on general-purpose hardware. Real-time clinical applications could greatly benefit from a hardware-accelerated data processing, especially in the case of FPGAs or Appliction Specific Integrated Circuits (ASICs) which are energy-efficient compared to traditional CPUs or GPUs, and can significantly reduce the computation time required to process large amounts of high-dimensional data [9] - [15].

In this paper, an FPGA-based spike sorting architecture is presented for real-time multi-channel clustering of neural data. This solution can help neuroscientists to position the electrode array before the real neural recording starts, so the offline classification efficiency can be improved. The Online Sorting (OSort) [16] algorithm is used, which original implementation works on only single-channel. Since neuroscientists require the correlation between the electrodes, so the original algorithm is modified to support multi-channel electrode arrays.

II. System Setup

The measurement is done with the RHD2000 electrophysiological recording system from Intan Technologies, which is an open-source hardware and software that allows users to record biopotential signals from up to 256 low-noise amplifier channels. The system includes two 32-channel RHD2132 and one 64-channel RHD2164 amplifier boards to do 128-channel neural recordings with 16-bit resolution. The 128-channel electrode array are built up from 4x32 electrodes, and the sampling frequency is 20kHz. The RHD2000 recording system already includes an Opal Kelly XEM6010 module, which contains a Spartan-6 FPGA board, and it could be utilized for online pre-clustering.

Using this setup a typical neural recording is 5 minutes long. However it is not rare to prepare several hours recording, if there are rarely firing neurons, which generate one spike in a second, or a day-night cycle monitoring is done on a specific neuron. The number of spikes are depends on the number of channels, the length of the recording, the detection threshold, the wakefulness of the patient, the chosen measurement area and the injury of the measured brain tissue, etc. In case of a five minute long measurements the number of detected spikes can be from 10 to 100 thousand. The expected number of clusters also depends on the earlier criterions, but in our experiments using a 128-channel setup the well-separable clusters are in the range of 15-35, with the average values of 20-25. In case of a longer neural recording these values can increase with 10-30%. In the proposed architecture only a 5x4 part of the 128 electrode is used, with 64 data points per spike.

III. Spike Detection

The first step of spike sorting is the detection of spikes. Based on [6] and [7], several spike detection algorithms have been implemented and tested in MATLAB. The results showed that the Non-Linear Energy Operator (NEO) is suitable for FPGA implementation [9], [10]. NEO is based on the following equation:

$$\psi[x(n)] = x(n)^2 - x(n+1)\cdot x(n-1), \tag{1}$$

where $x(n)$ is n^{th} sample of the original signal, and $\psi[x(n)]$ is the n^{th} sample of the NEO signal. The amplitude of the NEO signal is high only when the signal power and frequency are high. The detection based on the calculated NEO signal (1) and the detection threshold (T_D) calculated automatically as follows:

$$T_D = c_D \cdot \frac{1}{N}\sum_{n=1}^{N}\psi[x(n)], \tag{2}$$

where N is the number of samples and c_D is the detection correction factor, which is some power of two.

The spike detection architecture can be seen in Fig. 1. The data from the neural probe are interleaved by the $Serializer$ module and then stored in the external memory. According to (1) the NEO unit requires the n^{th}, $(n-1)^{th}$ and $(n+1)^{th}$ samples from each channel to calculate the corresponding NEO values. The n^{th} and $(n-1)^{th}$ data are temporarily stored in the BRAMs. The detection threshold value is computed in the $AVG\&Shift$ unit, based on (2). The $AVG\&Shift$ unit calculates the average of the NEO values, then shifts the result to left by the correction factor (in our case $c = 8$). Finally the $ThresholdModule$ reloads the NEO signal and compares it to the calculated detection threshold. If a NEO value is above the threshold the $ThresholdModule$ transfers the spike and all other corresponding samples of each channel (5x4) from the original signal. This spike data is transferred to the $SpikeMemory$ via AXI-Stream.

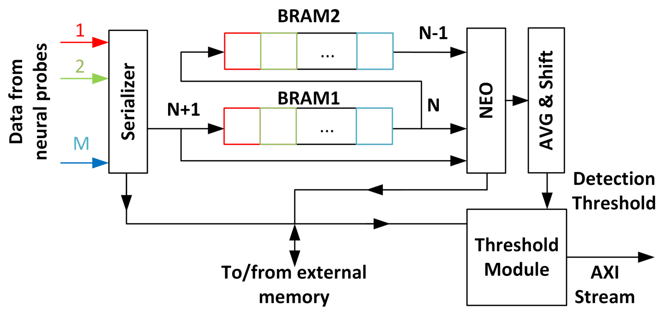

Fig. 1. The architecture of the NEO spike detection

The spike detection can be separated from the spike clustering, so it can be changed easily to another detection architecture for future experiments.

IV. Spike Sorting - OSort

The OSort [16] is an online unsupervised template-matching spike sorting algorithm, which works on the detected spikes and uses them as features. Important parts of the algorithm are the automatically computed clustering and merging thresholds (T_C,T_M), which are calculated based on the standard deviation of the signal as follows:

$$T_C = T_M = std(signal)^2 \cdot c_C \cdot N_S, \tag{3}$$

where c_C is the clustering correction factor (1.15 in our case), and N_S is the number of data points in a spike.

The algorithm works as follows: When the first spike is received from the detector, the mean of this spike is stored as the first cluster. The next incoming spike is compared to the cluster mean using squared difference as distance metric. If this distance is below the cluster threshold, then the spike is assigned to this cluster, if not then a new cluster is formed. This process is applied to all spikes. When the cluster assignment is done the mean of the cluster is updated and the merging begins. In this process the distance between the stored means and the new cluster mean are calculated. When the cluster with the lowest distance is below the merging threshold, then it is merged with the updated cluster.

V. Multi-channel OSort implementation

The original version of the OSort algorithm [16] works with a single-channel. To handle multi-channel measurements some modification is required. The modified version of the algorithm is similar to the original one, but the data flow and structure is completely redesigned. An FPGA-based version of the OSort algorithm was already reported in the literature [9], but due to the increased data requirements in the multi-channel case it runs into memory bandwidth problems. In our case one cluster has $5 \times 4 \times 64$ data points, and the spike

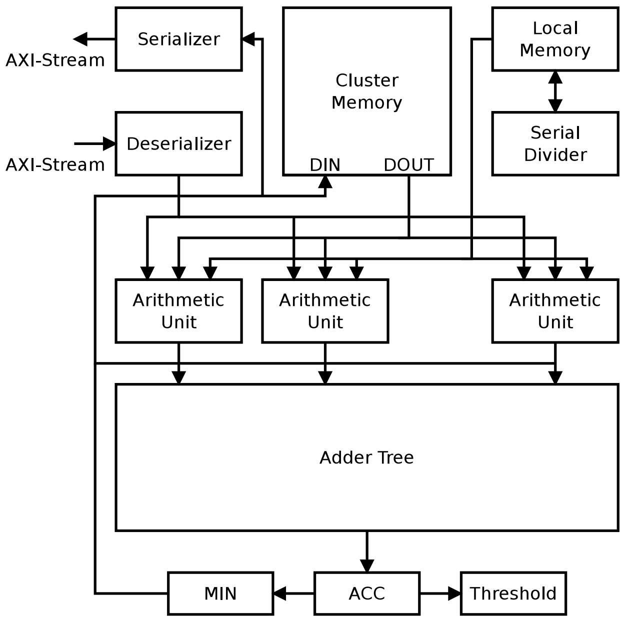

Fig. 2. The architecture of one Cluster Module

data is stored as 16 bit signed integer values and the cluster mean values are stored as 18 bit fixed point numbers with 2 bit fractional part. The original version of OSort is working with 64 or 256 (interpolated) samples per spike and use double precision for storage. Our experiments show that rounding the cluster means to integer values during the computation does not alter the resulting clusters significantly.

In the widest data bus configuration the 36 kbit Block-RAMs (BRAM) of the Xilinx 7-series FPGAs are 72 bit wide and 512 element deep therefore 8 clusters can be stored in 5 BRAMs. The expected number of clusters in our current application are in the order of 100, therefore in the initial implementation the number of clusters is maximized to 128 clusters, which can be stored in 80 BRAMs. Computations of the OSort algorithm is executed on an array of processors with 20 (5×4) Arithmetic Units where one multiplier and 3 adders are used in each unit. Therefore low-cost FPGAs can be suitable for this task.

The architecture of the Cluster Module is shown in Fig. 2. Spike data is loaded from an on-chip BRAM via the input AXI-Stream bus and go through an optional deserializer. The data width of the stream can be reduced by using serialization when multiple Cluster Modules are computing in parallel, but the order of computations must be changed in this case. For simplicity only the non-serialized operation is discussed in the current paper.

In the first stage of the OSort algorithm spike data for each channel is loaded in parallel through the AXI-Stream bus and the corresponding mean values of the first cluster is provided by the Cluster Memory. Partial results of the summed squared difference is computed by the Arithmetic Units. The partial results are summed by the Adder Tree for each sample and an accumulator (ACC) is used to compute the sum over the entire 64 elements sample window. The result and the cluster number is saved into the MIN register if it is smaller than the previous minimum squared distance. These steps should be executed for each living cluster in the Cluster Memory.

TABLE I. AREA REQUIREMENTS AND DEVICE UTILIZATION OF THE SYNTHESIZED CLUSTERING MODULE

	Resource requirements			
	FF	LUT	DSP	BRAM
	12,150	14,037	120	102
	Device Utilization			
	FF	LUT	DSP	BRAM
XC7Z7020	11.42%	26.38%	54.55%	72.86%
XC7Z7100	2.19%	5.05%	5.94%	18.72%
XC7K325T	2.98%	6.88%	14.29%	22.92%
XC7V690T	1.40%	3.24%	3.33%	6.94%

If the distance of the current spike between all clusters is computed and the value in the MIN register is larger than the T_C clustering threshold then the spike data is reloaded and saved into the Cluster Memory as a new cluster. If the distance is smaller than T_C the spike should be merged with the closest cluster. The weighted average of the cluster mean and the new spike is computed by the Arithmetic Units and the results are saved into the Cluster Memory and also to the spike memory. The weights which can be precomputed by the serial divider and stored in the local memory are $n/(n+1)$ and $1/(n+1)$ for the cluster mean and the new spike respectively, where n is the number of spikes in the cluster. The new number of spikes $(n+1)$ and the new weights $((n+1)/(n+2),\ 1/(n+2))$ is computed and stored in the local memory.

In the third stage the cluster means, which updated previously, is loaded from the spike memory and its distance is computed for all clusters similarly to stage one. When the computed minimum distance is smaller than the T_M merging threshold the mean values of the two clusters are merged in stage four similarly to stage two. If the recently updated cluster A has n elements and the closest cluster B has m elements then the weights are $n/(n+m)$ and $m/(n+m)$, which can be precomputed when the distance of the two clusters is computed in stage three. Cluster A marked as unused and merged in a table held in the local memory and the new number of spikes in cluster B is computed $(n+m)$. The new weights required in stage two $((n+m+1)/(n+m+2),\ 1/(n+m+2))$ are also precomputed. The system halts until a new spike is detected and computation is restarted from stage one.

VI. RESULTS

The proposed Clustering Module is developed using Vivado HLS 2016.4. The original MATLAB algorithm is translated to C/C++ taking into account the architecture described in Section V. and the special requirements of High Level Synthesis (HLS) compiler. The spike input and cluster number outputs are mapped to an AXI-Stream bus while the threshold can be set via an AXI-Light connection. The estimated dedicated resource requirements are 80 BRAMs to store the 128 clusters, 5 BRAMs to store spike data and 1 BRAM for the local memory. Each Arithmetic Unit requires one DSP slice and a 42 bit accumulator, therefore 20 DSP slices required in the 20 channel case. The full system requires additional blocks such as AXI interconnect, AXI-DMA engine, memory, Ethernet, ADC interface to connect to peripheral devices and a MicroBlaze or ARM processor in case of Zynq-7000 devices to control the high level operation of the system.

TABLE II. DETAILED LATENCY OF THE CLUSTERING MODULE

		Latency	Iteration Latency	Trip Count
Load Spike	load_spike_sp_ch	1,281	3	1280
Stage 1	comp_diff_sc	8,197	7	8,192
Stage 2	update_mean	66	4	64
	mean_save	64	2	64
	new_cluster	64	2	64
Stage 3	comp_diff_cc	8,197	7	8,192
Stage 4	merge_clust	66	4	64
	merge_save	64	2	64
Read merge table	merge_read	129	3	128

The real area requirements of the synthesized Clustering Module is shown on Table I. The memory and DSP requirements of the architecture synthesized by Vivado HLS are larger than expected. During synthesis the spike and cluster memories are build from BRAMs configured as 18 bit$\times$1024 and 2 bit$\times$8192 respectively. The DSP slice requirements are six times higher than our first estimation. It seems that DSP resources are not shared between the different stages of the algorithm when integer operands used during computation. The Arithmetic Unit of stage one and three requires 20-20 DSP slices while stage two and four is computed by using 40-40 DSP slices, despite these steps are executed serially. In spite of the suboptimal synthesis the proposed architecture still can be implemented on a Xilinx KC-705 development board with Kintex-7 XC7K325T FPGA. Even a cheap Zynq device (XC7Z7020) is suitable for the task.

Due to the relatively low operand bit width the system can operate on 200MHz clock frequency. If the number of clusters is maximized to 128 the first and third stage of the OSort algorithm can be computed in approximately 8192-8192 clock cycles (64 clock per cluster, 128 clusters). Computation of the second and fourth stages are divided into two parts: computation of the new mean values and saving them into the Cluster Memory which can be carried out in 130 clock cycles. Running on 200MHz clock frequency one spike ($5\times4\times64$ data points) can be clustered in 18,127 clock cycles in the worst case, which results in 90.635μs clustering time for a spike or more than 11,000 spike/s.

The proposed Clustering module was implemented on XC7Z7020 FPGA. Actual latencies of the different stages of the architecture operate on 101MHz clock frequency are summarized on Table II. The test results showed, that spikes can be clustered with an average of 18,005 clock cycles. Based on the results, it can be concluded that the estimations were right.

VII. CONCLUSION AND FUTURE WORK

In this paper an FPGA-based implementation of the OSort algorithm for unsupervised online multi-channel neural spike clustering is proposed. The results show that the presented architecture can be implemented on a mid range FPGA device running on 200MHz, which classifies the incoming spikes from 20 distinct channels in 90.635μs, or 11,000 spikes/s. Furthermore it can be concluded that it is 40 times faster than the identical algorithm running offline on the PC (i7-4770, 3.4GHz, 8GB DDR3) in MATLAB. For a fair comparison the MATLAB implementation was not compared to an optimized C/C++ based implementation. The proposed architecture currently sorts 20 channels of neural data in only one 5×4 window, but in the future we want to increase the number of channels and also the number of windows. Furthermore the windows could overlap with each other and calculated in parallel, so the architecture could be utilized to sort 128-channel neural recordings in real-time.

ACKNOWLEDGMENT

This research project is supported by KTIA_13_NAP-A-IV/1-4,6 for Istvan Ulbert.

REFERENCES

[1] G. Buzsaki, "Large-scale recording of neuronal ensembles," Nat Neurosci, vol. 7, pp. 446-51, May 2004.

[2] M. S. Lewicki, A review of methods for spike sorting: the detection and classification of neural action potentials, Network, vol. 9, pp. R53-78, Nov 1998.

[3] S. Fujisawa, A. Amarasingham, M. T. Harrison, and G. Buzsaki, Behavior-dependent short-term assembly dynamics in the medial prefrontal cortex, Nat Neurosci, vol. 11, pp. 823-33, Jul 2008.

[4] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, et al., Neuronal ensemble control of prosthetic devices by a human with tetraplegia, Nature, vol. 442, pp. 164-71, Jul 13 2006.

[5] D. M. Taylor, S. I. H. Tillery, and A. B. Schwartz, Direct cortical control of 3D neuroprosthetic devices, Science, vol. 296, no. 5574, pp. 18291832, June 2002.

[6] H. G. Rey, C. Pedreira, R. Q. Quiroga, Past, present and future of spike sorting techniques, Brain Research Bulletin, vol. 119, pp. 106117, 2015

[7] S. Gibson, J. W. Judy, D. Markovic, Comparison of Spike-Sorting Algorithms for Future Hardware Implementation, 30th Annual International Conference of the IEEE on Engineering in Medicine and Biology Society, 2008

[8] C. M. Lopez, A. Andrei, S. Mitra, M. Welkenhuysen, W. Eberle, C. Bartic, et al., "An Implantable 455-Active-Electrode 52-Channel CMOS Neural Probe," IEEE Journal of Solid-State Circuits, vol. 49, pp. 248-261, Jan 2014.

[9] S. Gibson, J. W. Judy, and D. Markovic, An FPGA-based platform for accelerated offline spike sorting, J Neurosci Methods, vol. 215, pp. 1-11, Apr 30 2013.

[10] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, et al., "Real-Time FPGA-Based Multichannel Spike Sorting Using Hebbian Eigenfilters," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 1, pp. 502-515, 2011.

[11] W. J. Hwang, W. H. Lee, S. J. Lin, and S. Y. Lai, "Efficient architecture for spike sorting in reconfigurable hardware," Sensors (Basel), vol. 13, pp. 14860-87, 2013.

[12] M. Pachitariu, N. A. Steinmetz, S. Kadir, M. Carandini and K. D. Harris, Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels, bioRxiv dx.doi.org/10.1101/061481, 2016

[13] L. Schaffer, Z. Nagy, Z. Kincses, Zs. Voroshazi, R. Fiath, I. Ulbert, and P. Szolgay, FPGA-based clustering of multi-channel neural spike trains, CNNA 2016, Dresden, Germany, August 23-25. 2016.

[14] Z. Jiang, Q. Wang, M. Seok, A low power unsupervised spike sorting accelerator, 52nd Design Automation Conference (DAC), 2015

[15] V. Karkare, S. Gibson, D. Markovic, A 75-W, 16-Channel Neural Spike-Sorting Processor With Unsupervised Clustering, IEEE Journal of Solid-State Circuits, volt. 48, issue 9, pp.2230-2238, 2013

[16] U. Rutishauser, E. M. Schuman, and A. N. Mamelak, Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo, J Neurosci Methods, vol. 154, pp. 204-24, Jun 30 2006.