Memory Efficient On-Line Streaming for Multichannel Spike Train Analysis

Bo Yu, Terrence Mak, Leslie Smith, Yihe Sun, Alex Yakovlev and Chi-Sang Poon

Abstract— **Rapid advances in multichannel neural signal recording technologies in recent years have spawned broad applications in neuro-prostheses and neuro-rehabilitation. The dramatic increase in data bandwidth and volume associated with multichannel recording requires a significant computational effort which presents major design challenges for brain-machine interface (BMI) system in terms of power dissipation and hardware area. In this paper, we present a streaming method for implementing real-time memory efficient neural signal processing hardware. This method exploits the pseudo-stationary property of neural signals and, thus, eliminates the need of temporal storage in batch-based processing. The proposed technique can significantly reduce memory size and dynamic power while effectively maintaining the accuracy of algorithms. The streaming kernel is robust when compared to the batch processing over a range of BMI benchmark algorithms. The advantages of the streaming kernel when implemented on field-programmable gate array (FPGA) devices are also demonstrated.**

I. INTRODUCTION

Brain-machine interface technology offers an exciting means to study and to communicate with the brain [1]. As one neurophysiological signal measuring technique, multi-electrode arrays (MEAs) provide extremely high resolution neuronal signals and enable studies on large neural network ensembles [2].

The large data bandwidth and data volume associated with MEA's recording requires a significant computational effort for data filtering and analysis. Most neural signal analysis algorithms are highly computationally expensive. As a result, software based approaches for multichannel neural signal analysis require off-line processing. Modern Field Programmable Gate Arrays (FPGAs) embedding massive hardware computing resources provide an ideal platform for sophisticated real-time neural signal processing and data mining for multichannel neural recording systems.

When designing FPGA based multi-channel recording systems, large size on-chip memories are always required to allow neural signal processing on a batch of neural data recorded by each channel. As the number of channel increases, the memory requirement grows drastically. In this paper, we present an online streaming kernel to mitigate the problem associated with large memory size requirement, and thus to make real-time processing of multi-electrode signals more practicable. The online streaming kernel exploits the pseudo-stationary characteristic of neural spikes and, thus eliminates the need for long-term storage in signal processing, leading to more efficient hardware implementation. This design principle is exemplified by several commonly used neural signal analysis algorithms. The accuracy of the stream-based kernels is evaluated by comparing them with their batch-based computational counterparts. The improvements in memory size and power consumption are rigorously evaluated using FPGA devices.

The paper is organized as follows: Section II introduces the principle of streaming method. Section III software and hardware evaluation results are presented and discussed.

II. STREAMING METHOD

A. Pseudo-stationary property

Neural signals recorded during *in vivo* experiments have non-stationary nature which is mainly due to the relative movements between the recording electrodes and the recorded neurons [3]. However, neural signal can be regarded as pseudo-stationary in considering of a short recording window and relatively stable conditions, simply because the chance of disturbance is small. Although this paper applies online streaming method on algorithms for stationary spike sorting and neural signal processing, this method still has chance to be used in conjunction with algorithms for non-stationary neural signal processing.

During a short period of time, in which pseudo-stationarity of neuronal signal exists, the shape of a neuronal spike stays relatively fixed and only can be disturbed by various noises. This similarity in recorded neuronal signals implies that the same operation performed on signals picked from different sub-windows can lead to similar results due to the similarity of signals. The physical memory can be reduced or removed, because similar data can still be obtained from the signal source again even though the original data is discarded. Our streaming method is based on this similarity of neural signals.

B. Streaming principle

Batch processing requires pre-storage of data to be processed. The same set of data is processed by all the operations. Because spike characters are similar under the assumption of pseudo-stationarity, operations can be employed on different data sets to yield approximated results. We refer the methods that need to store a batch of data for processing as

This work is supported by EPSRC Knowledge Transfer Account. Bo Yu and Yihe Sun are with Tsinghua National Laboratory for Information Science and Technology, Institute of Microelectronics, Tsinghua University, Beijing 100084, China. Terrence Mak and Alex Yakovlev are with School of Electrical, Electronic and Computer Engineering, Newcastle upon Tyne, NE1 7RU, UK. Terrence Mak is also with the Institute of Neuroscience, Newcastle Biomedicine, at the same University. Leslie Smith is with department of Computing Science and Mathematics University of Stirling, Stirling, FK9 4LA, UK. Chi-Sang Poon is with Harvard-MIT Division of Health Sciences and Technology, MIT, Cambridge, MA 02139, USA.

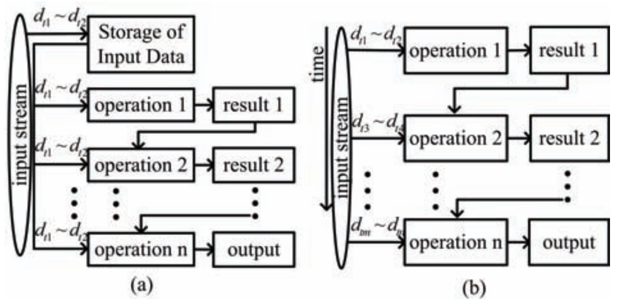

Fig. 1. (a) Batch based processing. (b) Stream based processing.

batch based methods in this paper. Taking the advantage of the pseudo-stationary property in neural recordings, streaming methods allow operations to be performed on the data sets piece by piece. As a result the need for a large memory is reduced.

Fig. 1 illustrates the differences between the streaming and batch based methods. In batch based processing, input data is stored for all the operations. Without storing input data, the streaming algorithm uses data from input directly and discards it when an operation finishes. The pseudo-stationary property implies that similar results can be obtained even using later data. As a result, the streaming method is a processing flow where minimal data storage is required.

C. Benchmark algorithms and hardware

Several neural signal processing algorithms including general Hebbian algorithm (GHA) [4], k-means algorithm, covariance calculation and Bell and Sejnowski algorithm [5] are taken as benchmarks for stream based software and hardware evaluation.

1) Stream based algorithm: General Hebbian algorithm (GHA) is an unsupervised neural network algorithm based on a certain form of auto-associative learning. Synaptic weights of the feed forward network in GHA evolves into principal components of input data if a specific weight updating rule, Oja's rule [4], is followed. As a result, GHA can be used for principal component analysis (PCA) that is a well-known spike feature extraction method.

The stream based GHA is shown in Table I. $\vec{x}(i) = [x_1(i), x_2(i), \ldots, x_m(i)]^T$ represents the ith $m \times 1$ input vector. $\vec{W}(j) = [\vec{W}_1(j), \vec{W}_2(j), \ldots, \vec{W}_l(j)]$ is the $l \times m$ synaptic weight matrix and initialized to $\vec{W}(1)$, "j" is the learning step. "η" is the learning rate and initialized to a small positive value. "$\vec{\mu}$" is the $m \times 1$ mean vector. $\vec{y}(j)$ is a $l \times 1$ vector. $LT[\vec{y}(j)\vec{y}^T(j)]$ sets all the elements in the matrix above the diagonal to zero. For a large learning step, j, $\vec{W}(j)$ converges to the first l principal components of the input data.

The difference between stream based GHA and batch based GHA is whether the same input vectors are used for both mean operation and zero-mean transformation. In batch based algorithm, the two operations are performed on the same data. Memories are required to cache these data for the two sequential operations. In stream based algorithm, data

TABLE I

ALGORITHM OF STREAM BASED GHA

```
1. Initialize synaptic weight $\vec{W}(1)$ and learning rate $\eta$.
   Select the number of data, N, for estimating mean value
2. Mean operation
   $\vec{\mu} = \sum_{i=1}^{N} \vec{x}(i)/N$
3. Zero-mean transformation
   $\vec{x}(i) = \vec{x}(i) - \vec{\mu} \quad i > N$
4. Perform learning on mean centered data
   $\vec{y}(j) = \vec{W}(j)\vec{x}(i)$
   $\vec{LT}(j) = LT[\vec{y}(j)\vec{y}^T(j)]$
   $d\vec{W}(j) = \eta(\vec{y}(j)\vec{x}^T(i) - \vec{LT}(j)\vec{W}(j))$
   $\vec{W}(j+1) = \vec{W}(j) + d\vec{W}(j)$
   $j = j+1, i = i+1$
5. If convergent is not achieved, go to step 3
   Otherwise stop algorithm.
```

Note: 1. $\vec{x}(i)$ is input vector, $\vec{W}(j)$ is synaptic weight, η is the learning rate, $LT[\cdot]$ sets all the elements in the matrix above the diagonal to zero. 2. In batch based algorithm mean operation and zero-mean transformation are performed on the same data i.e. $\{\vec{x}(i)\}, i \in [1, N]$.

used by mean operation is different from data used in zero-mean transformation. Notice that the stream based algorithm does not change the algorithm kernel of original algorithm. It only change the way of utilizing data.

For the similar reason to the general Hebbian algorithm, the other benchmark algorithms need to store a batch of input data. For these benchmarks, based on the pseudo-stationary property, the streaming algorithms can be obtained from their batch based counterparts.

2) Stream based hardware: Hardware Hebbian eigenfilter is taken as an example to illustrate the difference between stream based and batch based structure. Fig. 2 shows the structure of hardware Hebbian eigenfilter. It is an example where the first three principal components are filtered. It consists of "learning kernel", "system controller", "mean calculator", "interface" and "memory". Memory block only exists in batch based structure. "System controller" controls the operation of the whole system. "Learning kernel" performs learning operations and consists of arithmetic units, storing units and switchers. "LT" stores the result of $\vec{LT} = LT[\vec{y}\vec{y}^T]$, "score" stores result of $\vec{y} = \vec{W}\vec{x}$, "$Weight_{1,2,3}$" stores the three synaptic weights. "Mean calculator" calculates mean of data before mean is ready. After mean is ready, "mean calculator" subtracts mean from data and sends mean centered data to "learning kernal". In batch based Hebbian eigenfilter, "interface" write the input streaming data to the memory and read data from memory for mean calculation and Hebbian learning. In stream based Hebbian eigenfilter, "interface" directly forwards the input data to "mean calculator".

III. RESULTS

A. Evaluation Methodology

Both clinical spike trains [6] and synthetic spike trains are used for evaluation. In this paper, synthetic spike trains are generated through the spike time generating tool [7] and the spike train synthesis tool [8]. The spike time generating tool is used to generate neurons' firing times through specifying

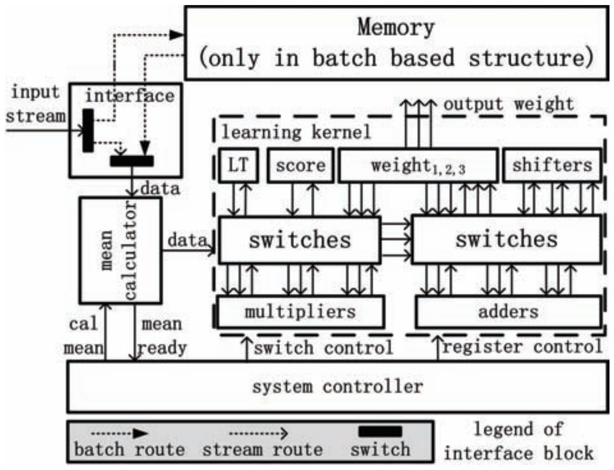

Fig. 2. The structure of Hebbian eigenfilter.

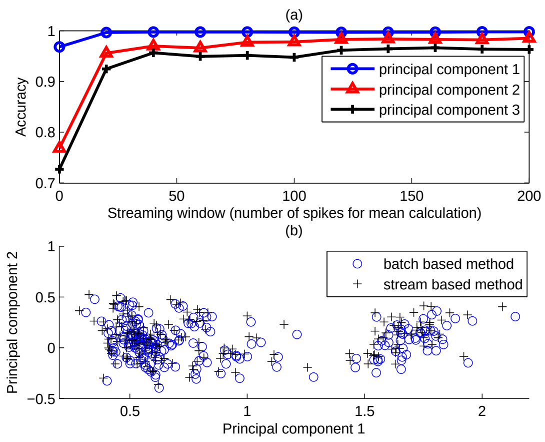

Fig. 3. (a) Accuracy of stream based Hebbian eigenfilter for spike sorting. (b) Projection score in principal components space (with streaming window of 200).

number of neurons, each neuron's firing rate and correlations among these neurons. Using generated spike firing times, the spike train synthesis tool generates noisy spike train through specifying the number of neurons, neuronal spike shapes and signal to noise ratio of spike train.

B. *Stream based algorithm evaluation*

1) The general Hebbian algorithm: Clinical data [6] is used to evaluate the stream based general Hebbian algorithm. In stream based routine, we use M aligned spikes to calculate the mean vector. The following N spikes are used for iteration learning and batch based general Hebbian algorithm. In principal component analysis, the direction of each principal component affects the projection results. Generally, the first three principal components capture most variances in the data. As a result, the accuracies of the first three principal components are defined as,

$$\text{accuracy} = |\vec{PC}_{i,stream} \cdot \vec{PC}_{i,batch}| \quad i = 1,2,3 \qquad (1)$$

where $\vec{PC}_i$ represents the ith principal component. A value of 1 represents that stream and batch based principal components have identical direction. Fig. 3(a) shows the relationship between the number of spikes for mean calculation in streaming calculation and the accuracy of the stream based algorithm. The accuracy increases with the growth of the streaming window.

Fig. 3(b) shows projection results in principal component space using both Matlab function (princomp, which is batch based) and our streaming Hebbian eigenfilter. In stream based method, the streaming window size is 200.

2) Bell and Sejnowsk algorithm: In the stream based algorithm, we use M samples for mean calculation and the following N (N is streaming window) samples for iteration learning defined in the algorithm. The batch based algorithm is performed on the same N samples as the streaming algorithm.

N synthetic spike trains, $\vec{X}$, are generated through the spike train synthesis tool. The spike trains in $\vec{X}$ are mixed by an $N \times N$ matrix $\vec{W}$ (N is 3 in this evaluation). An un-mixing matrix, $\vec{B}$, is the result of Bell and Sejnowski algorithm. If un-mixing matrix $\vec{B}$ can accurately infer source signals, $\frac{\vec{B}_i \cdot \vec{W}}{|\vec{B}_i \cdot \vec{W}|}$ should have the same direction of a unit vector, $\vec{u}$, where $\vec{B}_i$ is a row vector of matrix $\vec{B}$, $\vec{u}$ is $[100]^T$ or $[010]^T$ or $[001]^T$. We use $\frac{\vec{B}_i \cdot \vec{W}}{|\vec{B}_i \cdot \vec{W}|} \cdot \vec{u}$ to evaluate accuracy. With a correct un-mixing weight matrix, this value equals one. Fig. 4 shows the accuracy of both streaming and batch processing methods. The stream based algorithm can finally get the un-mixing weight matrix if the number of iteration steps is large enough.

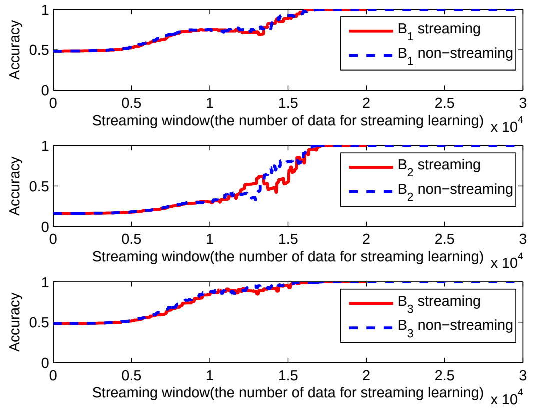

Fig. 4. Accuracy of streaming Bell and Sejnowski algorithm.

3) Covariance matrix calculation: In stream based covariance calculation, N (N is a variable) aligned spikes are used for mean calculation. Using mean value, other M spikes (M is 1024 in our experiment) are used to calculate covariance matrix. Clinical data is used to evaluate the streaming method. Results in Fig. 5(a) compare the covariance matrix computed by streaming method and batch processing. The error is the root mean square error of differences between the streaming and batch processing covariance matrix.

TABLE II

AREA AND PERFORMANCE OF STREAMING AND BATCH PROCESSING METHOD

		covariance matrix(10 bits)		Hebbian eigenfilter(10 bits)		k-means(10 bits)		ICA(32 bits)	
Device	Parameter	stream	batch	stream	batch	stream	batch	stream	batch
Virtex6	Clock Freq.(MHz)	50	50	50	50	50	50	50	50
	Power(mW)	5.8	12.6	5.4	11.3	7.2	10.5	43.6	52.3
	Logic cost	142	217	496	565	2071	2106	19291	19288
	Memory cost	5	23	5	23	0	3	0	6
Spartan6	Clock Freq.(MHz)	50	50	50	50	50	50	50	50
	Power(mW)	5.1	7.4	2.4	7.3	14.3	15.6	111	122.8
	Logic cost	142	291	488	628	2073	2117	19173	19175
	Memory cost	6	42	5	41	0	3	0	12

note: Logic cost and memory cost are consumed LUT (look-up table, basic logic elements in FPGA) and BRAM (basic memory block in FPGA).

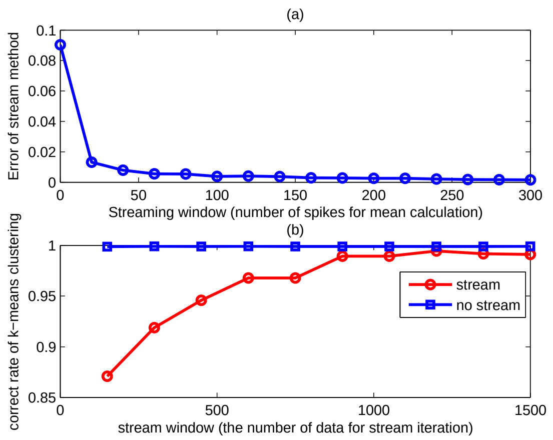

Fig. 5. Accuracy of stream based covariance matrix (a) and k-means algorithm (b).

4) K-means algorithm: Matlab function *princomp* projects aligned spikes generated by spike synthetic tool into feature space formed by the first two principal components. The k-means algorithm is performed on the data in the feature space. In our streaming k-means, we use N spike points for each iteration. Centroids are updated after each iteratoin. To avoid obtaining a local optimized results that can lead to obvious clustering error, the centroids are re-calculated every M (M is 3 in our experiment) iterations. We use the correct rate of clustering to evaluate the accuracy of the streaming method. The correct rate is defined as,

$$\text{correct rate} = \frac{\text{Num}_{\text{correct\_classified\_spikes}}}{\text{Num}_{\text{total\_spikes}}} \qquad (2)$$

where $\text{Num}_{\text{correct\_classified\_spikes}}$ is the number of correctly classified spikes and $\text{Num}_{\text{total\_spikes}}$ is the total number of spikes. Fig. 5(b) shows the relationship between correct rate and the streaming window size in streaming method. In the streaming method, the more data used for streaming, the more re-calculations can be done, so correct rate of k-means clustering increases as the increases in the streaming data. When streaming window is larger than 1000, the streaming k-means algorithm achieves a relatively high correct rate (around 0.99).

C. Stream based hardware

FPGA is used for evaluating memory and power reduction on the benchmark hardware through utilizing streaming method. Both stream based and batch based benchmark algorithms are designed. Xilinx System Generator is the hardware design tool. We employ Xilinx-ISE and Xilinx XPower to obtain the resource and power estimation of the design. Table II lists power and hardware resource. From the result we can see that 16.6% to 54% power consumption can be reduced by using our streaming method if implementing algorithms on Virtex6, and 8.3% to 67% power can saved if implementing algorithms on Spartan6. BRAMs usage in all implementations can also be greatly reduced by using our streaming approach.

IV. CONCLUSIONS

This paper presents a streaming method for analyzing neural recordings from multi-electrode arrays. Several neural signal analysis algorithms are adopted to exemplify the design methodology and are implemented on FPGAs. Using the streaming method, FPGA's block memory usage and power dissipation can be largely reduced when compared to conventional batch processing. In the future, we will explore the application of streaming design principle for developing bi-directional brain-machine-interface systems.

REFERENCES

[1] M. Lebedev and M. Nicolelis, "Brain-machine interfaces: past, present and future.," *Trends in Neurosciences*, vol. 29, no. 9, pp. 536–546, 2006.

[2] U. Frey, U. Egert, F. Heer, S. Hafizovic, and A. Hierlemann, "Microelectronic system for high-resolution mapping of ectracellular electric fields applied to brain slices.," *Biosensors and Bioelectronics*, vol. 24, pp. 2191–2198, 2009.

[3] A. Bar-Hillel, A. Spiro, and E. Stark, "Spike sorting: Bayesian clustering of non-stationary data," *Journal of Neuroscience Methods*, vol. 157, pp. 303–316, 2006.

[4] S. Simon, "Neural networks and learning machines, 3rd edition," pp. 395–446, 2009.

[5] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 6, pp. 1151–1155, 1995.

[6] R. Quiroga, "Wave clus." `http://www2.le.ac.uk/departments/engineering/research/bioengineering/neuroengineering-lab/spike-sorting.htm`, July 2009.

[7] J. Macke, P. Berens, A. Tolias, and M. Bethge, "Generating spike trains with specified correlation coefficients," *Neural Computation*, vol. 21, pp. 397–423, 2009.

[8] L. Smith and N. Mtetwa, "A tool for synthesizing spike trains with realistic interference," *Journal Neuroscience Methods*, vol. 159, pp. 170–180, 2007.