

Efficient VLSI Architecture for Spike Sorting Based on Generalized Hebbian Algorithm

Wen-Jyi Hwang and Hao Chen

Department of Computer Science and Information Engineering,
National Taiwan Normal University, Taipei, 117, Taiwan.

Abstract.

A novel hardware architecture for fast spike sorting is presented in this paper. The architecture is able to perform feature extraction based on the Generalized Hebbian Algorithm (GHA). The employment of GHA allows efficient computation of principal components for subsequent clustering and classification operations. The hardware implementations of GHA features high throughput and low area costs. The proposed architecture is implemented by Field Programmable Gate Array (FPGA). It is embedded in a System-On-Programmable-Chip(SOPC) platform for performance measurement. Experimental results show that the proposed architecture is an efficient spike sorting design for attaining low hardware resource utilization and high speed computation.

1 Introduction

Spike sorting [1] is often desired for the design of brain machine interface (BMI). It receives spike trains from extracellular recording systems. Each spike train obtained from the system is a mixture of the trains from neurons near the recording electrodes. The goal of spike sorting is to segregate the spike trains of individual neurons from this mixture. Spike sorting is a difficult task due to the presence background noise and the interferences among neurons in a local area. A typical spike sorting algorithm involves computationally demanding operations such as feature extraction. One way to carry out these complex tasks is to deliver spike trains to external computers. Because the delivery of raw spike trains requires high bandwidth, wireless transmission may be difficult. Many existing spike sorting systems are therefore wired, restraining patients and test subjects from free movement.

Hardware spike sorting is an effective alternative for BMI applications. It allows the spike sorting to be carried out at the front-end so that data bandwidth can be reduced for wireless communication. A number of hardware architectures [2, 3, 4] have been proposed to expedite the spike sorting. A common drawback of these approaches is that they are not based on efficient pipeline operations. The architectures therefore may not be suitable for applications requiring online training of large number of channels.

The objective of this paper is to present an effective VLSI architecture for spike sorting. The architecture is able to perform online training for feature extraction in hardware. The feature extraction is based on principal component analysis (PCA), which is carried out by the generalized Hebbian algorithm

(GHA) [5]. A novel pipeline is proposed to implement the architecture. The number of stages in the pipeline is the same as the number of the principal components to be computed for PCA. Each stage is responsible for the computation of a principal component. In addition, the computation results of the precedent stages can be used for the computation of subsequent stages to accelerate the training process.

To physically evaluate the proposed architecture, a spike sorting system on a System-On-Programmable-Chip (SOPC) platform is implemented, where the proposed spike sorting architecture is used as a hardware accelerator. The soft-core processor in the SOPC platform does not participate the spike sorting computation. It is used for control and data delivery among different components in the SOPC. The computation time of spike sorting based on the SOPC is measured and compared with existing works. Experimental results reveal that the proposed architecture is able to perform feature extraction in real time with low hardware resource consumption.

2 Preliminaries

Let $\mathbf{x}(n) = [x_1(n), \dots, x_m(n)]^T$, and $\mathbf{y}(n) = [y_1(n), \dots, y_p(n)]^T$, $n = 1, \dots, t$, be the n -th input and output vectors to the GHA, respectively. In addition, m , p and t are the vector dimension, the number of principal components (PCs), and the number of input and output vectors for the GHA, respectively. The output vector $\mathbf{y}(n)$ is related to the input vector $\mathbf{x}(n)$ by

$$y_j(n) = \sum_{i=1}^m w_{ji}(n)x_i(n) \quad (1)$$

where the $w_{ji}(n)$ stands for the weight from the i -th synapse to the j -th neuron at iteration n .

Let $\mathbf{w}_j(n) = [w_{j1}(n), \dots, w_{jm}(n)]^T$, $j = 1, \dots, p$, be the j -th synaptic weight vector. Each synaptic weight vector $\mathbf{w}_j(n)$ is adapted by the Hebbian learning rule:

$$w_{ji}(n+1) = w_{ji}(n) + \eta y_j(n)[x_i(n) - \sum_{k=1}^j w_{ki}(n)y_k(n)], \quad (2)$$

where η denotes the learning rate. After a large number of iterative computation and adaptation, $\mathbf{w}_j(n)$ will asymptotically approach to the eigenvector associated with the j -th eigenvalue λ_j of the covariance matrix of input vectors, where $\lambda_1 > \lambda_2 > \dots > \lambda_p$. A more detailed discussion of GHA can be found in [5].

When GHA is used for spike sorting, the $\mathbf{x}(n)$ is then the n -th spike in the spike train. Therefore, the vector dimension m is the number of samples in a spike. Let $\mathbf{w}_j = [w_{j1}, \dots, w_{jm}]^T$, $j = 1, \dots, p$, be the synaptic weight vectors of the GHA after the training process has completed. Based on \mathbf{w}_j , $j = 1, \dots, p$, the GHA feature vector extracted from training vector $\mathbf{x}(n)$ (denoted by \mathbf{f}_n) is

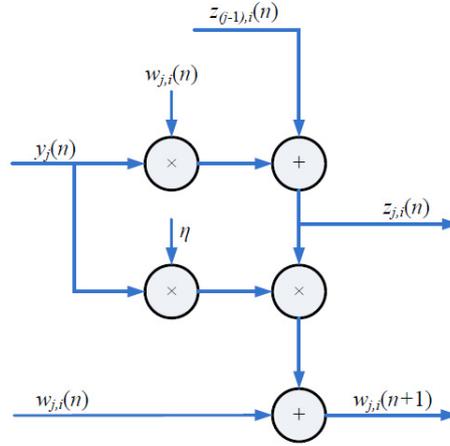


Fig. 1: The architecture for the implementation of eqs. (4) and (6).

computed by $\mathbf{f}_n = [f_{n,1}, \dots, f_{n,p}]^T$, where

$$f_{n,j} = \sum_{i=1}^m w_{ji} x_i(n) \quad (3)$$

be the j -th element of \mathbf{f}_n . The classification of the spike $\mathbf{x}(n)$ can then be based on the feature vector \mathbf{f}_n .

3 The Proposed Architecture

The goal of the proposed design is to implement eqs.(1) and (2) for GHA training in hardware. The hardware circuit for eq.(1) is termed the principal component computation (PCC) unit. It contains m multipliers and one adder. The circuit for eq.(2) is termed the synaptic weight vector updating (SWU) unit. Although the direct implementation of eq. (2) is possible, it will consume large hardware resources [6]. One way to reduce the resource consumption is by observing that eq. (2) can be rewritten as

$$w_{ji}(n+1) = w_{ji}(n) + \eta y_j(n) z_{ji}(n), \quad (4)$$

where

$$z_{ji}(n) = x_i(n) - \sum_{k=1}^j w_{ki}(n) y_k(n), j = 1, \dots, p. \quad (5)$$

and $\mathbf{z}_j(n) = [z_{j1}(n), \dots, z_{jm}(n)]^T$. The $z_{ji}(n)$ can be obtained from $z_{(j-1)i}(n)$ by

$$z_{ji}(n) = z_{(j-1)i}(n) - w_{ji}(n) y_j(n), j = 2, \dots, p \quad (6)$$

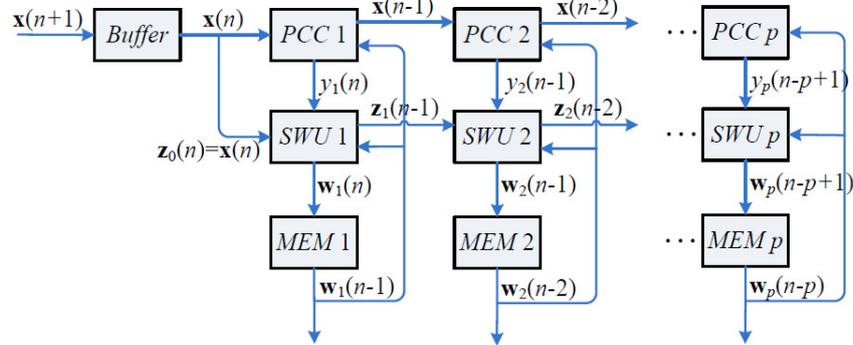


Fig. 2: The architecture of the proposed pipeline for GHA training.

When $j = 1$, from eqs (5) and (6), it follows that

$$z_{0i}(n) = x_i(n) \quad (7)$$

Therefore, the hardware implementation of eqs. (4) and (6) is equivalent to that of eq.(2). Figure 1 shows the hardware architecture for the implementation of eqs. (4) and (6).

Figure 2 shows the architecture of p -stage pipeline for GHA training, which produces p principal components for feature extraction. As shown in the figure, each stage contains a PCC unit, a SWU unit and a memory unit. Spikes are delivered to the pipeline one at a time. The PCC unit of stage j , $j = 1, \dots, p$, (denoted by PCC_j) receives the spike $\mathbf{x}(n - j + 1)$ for the computation of $y_j(n - j + 1)$ in time interval n . In addition, the SWU unit at stage j , $j = 1, \dots, p$, (denoted by SWU_j) produces the synaptic weight vector $\mathbf{w}_j(n - j + 1)$ in the same time interval. The computation of $\mathbf{w}_j(n - j + 1)$ requires the $\mathbf{z}_{j-1}(n - j + 1)$, $\mathbf{y}_j(n - j + 1)$ and $\mathbf{w}_j(n - j)$ as inputs. In addition to $\mathbf{w}_j(n - j + 1)$, the SWU unit also produces $\mathbf{z}_j(n - j + 1)$, which will then be used for the computation of $\mathbf{w}_{j+1}(n - j + 1)$. The memory unit at stage j (denoted by MEM_j) stores synaptic weight vector $\mathbf{w}_j(n - j)$. After the training process is completed, the synaptic weight vectors stored in the memory units are then used for the computation of feature vectors using eq. (3).

The proposed GHA circuit is used as a custom user logic in a SOPC platform consisting of softcore NIOS CPU, DMA controller and on-chip RAM,. All the spikes are stored in the on-chip RAM and then transported to the proposed GHA circuit for feature extraction. The DMA-based training data delivery is performed so that the memory access overhead can be minimized. The softcore NIOS CPU runs on a simple software for circuit activation and data delivery. It does not involve GHA computations.

Table 1: The CSR of the spike sorting system based on the proposed GHA circuit.

c	2	3	4
SNR=10 dB	99.6 %	95.8 %	92.7 %
SNR=20 dB	99.6 %	96.4 %	94.1 %

Table 2: The hardware costs of the proposed GHA circuit and entire SOPC.

	Logic Elements	Embedded Multipliers	Memory Bits
Proposed GHA	9144/149760	432/720	63488/6635520
Arch.	(6.10 %)	(60.00 %)	(0.96 %)
Entire	19734/149760	436/720	744854/6635520
SOPC	(13.18 %)	(60.56 %)	(11.23 %)

4 Experimental Results

The design platform for the experiments is Altera Quartus II with SOPC Builder and NIOS II IDE. The target FPGA device is Altera Cyclone IV EP4CGX150. In order to evaluate the performance of the proposed architecture for spike sorting, the simulator developed in [7] is adopted to generate extracellular recordings. The simulation gives access to ground truth about spiking activity in the recording and thereby facilitates a quantitative assessment of architecture performance. All the spikes are recorded with sampling rate 13500 samples/sec. The length of each spike is 2.67 ms. Therefore, each spike has 36 samples (i.e. $m = 36$). The number of PCs is $p = 2$ for the circuit design.

Table 1 shows the classification success rate (CSR) of the spike sorting system for various number of target neurons c and SNR levels. The CSR for spike sorting is defined as the number of spikes which are correctly classified by the total number of spikes. The feature vectors produced by the proposed GHA circuit are clustered by the fuzzy c -means (FCM) algorithm for the CSR measurement. It can be observed from Table 1 that the proposed circuit is able to achieve CCRs above 92 % for $c = 4$ with SNR=10 dB.

The hardware costs of the proposed architecture and entire SOPC are revealed in Table 2. There are three different area costs considered in the experiments: logic elements, embedded multipliers, and memory bits. It can be observed from the table that only small percentages of logic elements and memory bits available in the target FPGA are consumed by the circuit. In addition, the employment of SOPC does not significantly increase the hardware resource consumption.

In Table 3, we compare the computation time of the proposed GHA circuit with those of other hardware implementations [2, 3, 4] for feature extraction. Note that the implementation in [4] is based on ASIC. Its performances are normalized by [3] for the FPGA-based comparisons. The computation time of [4] in the table are the normalized ones reported in [3]. It can be concluded from

Table 3: Comparisons of the proposed GHA circuit with other FPGA-based feature extraction implementations.

	Proposed GHA Arch.	GHA Arch. in [2]	GHA Arch. in [3]	PCA Arch. in [4]
Clock Rate	50 MHz	70 MHz		
Comput. Time	2.91 ms	5.03 ms	5.6 ms	41.8 ms ^a
Target Device	FPGA Cyclone IV EP4CGX150	FPGA Virtex 6 XC6V SX315T	FPGA Spartan 6 XC6SLX150L	ASIC CMOS 0.35 μ m

^a Computation time equivalent to FPGA.

Table 3 that the proposed architecture has lower computation time than that of the other hardware implementations [2, 3, 4]. In fact, because the computation time of the proposed architecture is only 2.91 ms, the circuit is able to perform the feature extraction for 12381 channels per minute.

5 Concluding Remarks

The proposed architecture has been implemented by FPGA for physical performance measurement. The architecture is used as an hardware accelerator to the NIOS CPU in a SOPC platform. Experimental results reveal that the proposed spike sorting architecture has advantages of high CSR and high computation speed. For SNR=10, its CSR is above 92 % for four target neurons. Its GHA circuit has higher computation speed as compared with existing hardware GHA implementations. These results show that the proposed system implemented by FPGA is an effective realtime device for spike sorting at the front.

References

- [1] M.S. Lewicki, A review of methods for spike sorting: the detection and classification of neural action potentials, *Network Computer Neural System*, Vol. 9, 1998.
- [2] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C.-S. Poon, A Reconfigurable Hebbian Eigenfilter for Neurophysiological Spike Train Analysis, Proc. International Conference on Field Programmable Logic and Applications, pp.556-561, 2010.
- [3] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C.-S. Poon, 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.
- [4] T. Chen, K. Chen, Z. Yang, K. Cockerham, and W. Liu, A biomedical multiprocessor SoC for close-loop neuroprosthetic application, in International Solid-State Circuits Conference, pp. 434-435, 2009.
- [5] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., Pearson, New Jersey, 2009.
- [6] S. -J. Lin, W. -J. Hwang, and W. -H. Lee, FPGA Implementation of Generalized Hebbian Algorithm for Texture Classification, *Sensors*, Vol. 12, pp.6244-6268, 2012.
- [7] L. S. Smith and N. Mtetwa, "A tool for synthesizing spike trains with realistic interference," Vol. 159, pp.170-180, *Journal of Neuroscience Methods*, 2007.