# VLSI Design and Implementation of Density-based Spike Classification for Neuroprosthetic Applications

Li-Fang CHENG, Tung-Chien CHEN and Liang-Gee CHEN
National Taiwan University, Taipei
E-mail: {fannycube, djchen, lgchen}@video.ee.ntu.edu.tw

***Abstract*—Successful proof-of-concept laboratory experiments on cortically-controlled brain computer interface motivate continued development for neural prosthetic microsystems (NPMs). In order to improve the NPMs, one of the main issue is to realize the realtime spike sorting processors (SSPs). The SSP detects the spikes, extracts the features, and then performs the classification algorithm to differentiate the spikes for different firing neurons. Several architectures have been designed for the spike detection and feature extraction. However, the hardware for classification is missing. To complete the SSP, a density-based hardware-oriented clustering algorithm is adapted for the hardware implementation for the classification. In the hardware architecture level, the concepts of convolution and data reuse are adapted to further reduce the power consumption. The final implementation achieves 32.6 $\mu$W and 0.25 mm$^2$ in 90nm low-leakage process.**

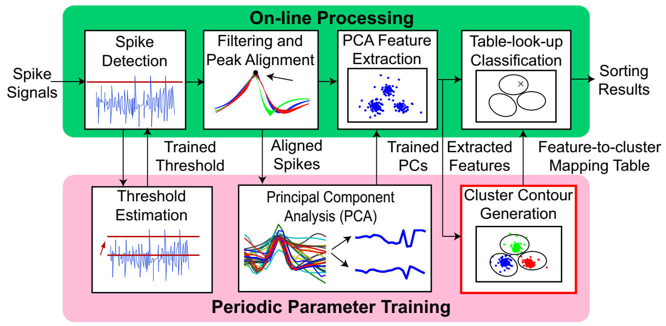
On-line Processing
Spike Signals
Spike Detection
Filtering and Peak Alignment
PCA Feature Extraction
Table-look-up Classification
Sorting Results
Trained Threshold
Aligned Spikes
Trained PCs
Extracted Features
Feature-to-cluster Mapping Table
Threshold Estimation
Principal Component Analysis (PCA)
Cluster Contour Generation
Periodic Parameter Training

Fig. 1. The hardware operation of spike sorting processors. We will focus on the the hardware design of the training device for the classification stage.

## I. Introduction

Spike sorting is an important tool for analyzing neural signals in the realm of neuroscience. It aims to sort the detected neural activities, or spikes, to the corresponding firing neurons. The performance of the cortically-controlled brain-machine interface for paralyzed patients may be improved with the aid of accurate sorting results [1]. One of the current research directions is to design a real-time spike sorting processor on the neural recording microsystems for the long-term experiments [2]. The power consumption and the size of the device are two of the major concerns for the hardware optimization.

The on-chip spike sorting system is generally composed of four stages: the spike detection, the filtering and alignment, the feature extraction and the classification. Each stage of the system can be divided into the on-line processing engine and the algorithm training engine. The training engine can collect a considerable amount of neural signals and extract the algorithm parameters used for the on-line processing engine. Many on-line processing hardware units for spike sorting have been proposed [3], [4]. The principal component analysis, one of the training algorithms for the spike feature extraction, has also been designed in VLSI hardware [5]. However in the previous works, the hardware for the training part of classification has not yet been proposed.

In this paper, we aim to implement the hardware of the training engine for classification. We first choose a density-based hardware-oriented algorithm proposed in [6] for the hardware design. In the hardware architecture level, we try to further reduce the power consumption of the device due to the large amount of data access power. The rest of this paper is organized as follow. In section II, we introduce some preliminary knowledge about the hardware design of spike sorting. Section III gives a review of the adapted hardware-oriented algorithm in this design. The proposed hardware architecture is represented in section IV. Section V shows the implementation results, and section VI gives the conclusion.

## II. Preliminary

### A. Spike Sorting

Neurons in brain communicate with each other through the firing of action potentials, or the so-called spikes. These spikes can be detected and recorded by extracellular micro-electrodes implemented in the brain. Many neuroscientific studies and applications require the measurements of the spikes for further improvement. However, the measured signals are often composed of multiple spikes from a group of close-by neurons, and how to analyze and identify the signals from different neurons accurately becomes an important issue. Spike sorting is the process to classify the detected spikes to their corresponding source neurons.

Figure 1 shows a general architecture of the hardware operation for spike sorting processors. The raw neural signals after the amplification and digitization are passed to the spike sorting processor. There are four major steps in the processor: the spike detection, the filtering and alignment, the feature

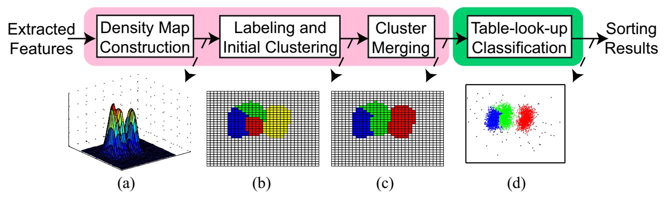

Fig. 2. The block diagram of the density-based clustering algorithm.

extraction, and the classification. Each stage can be further divided into the on-line processing part and the training part. The on-line processing engines deal with the sequential input signals in real time. The training engines compute the parameters for the on-line processing engines by collecting a considerable amount of neural data and implementing the training algorithms. In this paper, we will focus on the training engine in the classification stage for the spike sorting processor. The training engine for classification can read the spike features extracted by the feature extraction device and implement the clustering algorithms to classify the feature space. The mapping information of the feature space, such as a feature-to-cluster table, can then be generated and return to the on-line processing engine. The on-line processing engine is then able to classify the detected spikes in real time by table-look-up classification.

### B. *Requirements for the Hardware Implementation*

Note that the power consumption and the area are two major concerns for the spike sorting processors. In order to implement the hardware for the training engine of classification, a hardware-friendly algorithm should be chosen at first. There have been several algorithms used for the classification in the off-line spike sorting, such as k-means algorithm and mean shift algorithm [7], [8]. However, due to the large amount of required memory and computation complexity, most of them are not very feasible for hardware implementation. In this work, we choose a density-based hardware-oriented algorithm proposed in [6] for the hardware design because of the reported smaller memory usage and computation complexity in comparison with other traditional algorithms. The procedures of the algorithm will be introduced in the next section.

## III. Review of the Hardware-Oriented Algorithm for Classification

In this section, we will briefly review the procedure of the density-based hardware-oriented algorithm adapted in this work, and discuss its feasibility for hardware implementation.

### A. *Algorithm Procedure*

Figure 2 shows the overall procedure of the density-based hardware-oriented algorithm for classification [6]. In this algorithm, the concept of density estimation is adopted for the proved better performance of the mean shift algorithm than the traditional k-means algorithm. There are three steps in the algorithm. The first step of the algorithm is to create the density information of the extracted feature spaces. In order to calculate and store the density efficiently, the d-dimensional feature space is previously quantized to $N^d$ basic units, or the cells. $N$ is the level of quantization along each dimension. Next, a discrete symmetric profiling kernel is chosen for density accumulation. When each feature comes in, we first look for which cell contains the location of the feature, and then add the kernel weights to the cell and its adjacent cells. Through this accumulating procedure, we can get a density map like Fig. 2 (a) after all the spike features are read.

The second step of the algorithm is to cluster the cells in the feature space according to the density map created in the first step. For each cell, we compare its density with that of the adjacent cells, and a shift vector that indicates the direction to the adjacent cell with the highest density value is generated. We then group the current cell and the cell that the shift vector points to together, give them the same cluster label, and move on to the next cell for comparison. This iterative shifting and labeling procedure will be terminated when the cell with the local maximum of density is found. Start from different cells and repeat the procedure described above, we can classify the whole feature space. The initial clustering process is finished when all the cells are checked at least once. A cell-to-cluster table as shown in Fig. 2 (b) will be generated after this step. Finally, in the merging step, we check some merging conditions for the cells on the boundary of different clusters and merge these clusters if the criterions of the conditions are met. Fig. 2 (c) illustrates the cell-to-cluster table after the merging process, which is the final results of the classification. This table can be further transferred to the on-line classification device for table-look-up mapping. The re-mapping results of the on-line processing engine is shown as Fig. 2 (d).

### B. *Feasibility for Hardware Implementation*

The power and area are two main issues for the hardware implementation. In the algorithm level, we can use the memory usage and the computation complexity of the algorithm to estimate the feasibility for hardware realization. This is because the memory usage often contributes to most of the area, and the computation complexity can reflect the dynamic power consumption of a hardware device. Therefore, a good algorithm for hardware implementation should have the characteristics of low memory usage and computation complexity.

In comparison with the the traditional mean shift and k-means algorithm, the density-based algorithm introduced above reports a smaller amount of required memory and lower computation complexity. One reason is that since the spike features are transformed into the density domain with quantization before the clustering procedure, the memory usage can be reduced. Besides, in both k-means and mean shift algorithm, a large amount of of distance calculation is adopted during the iterative clustering procedure. However, in the density-based algorithm, only comparisons between the density of the cells are required. Since the comparison is simpler to realize than the distance calculation for hardware

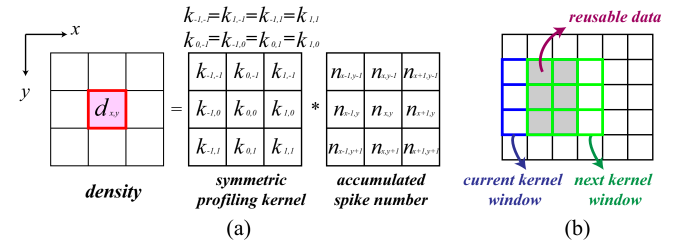

Fig. 4. (a) The procedure of the convolution method. (b) The data reuse scheme used during the procedure of convolution.

devices, the computation reduction and thus power saving of the device can be expected. For more details about the comparison results, please refer to [6]. Therefore, the density-based algorithm should be more feasible for the hardware design of the classification training device.

## IV. VLSI Architecture Design

In this section, the hardware architecture design based on the density-based classification algorithm discussed in section III is proposed. We first introduce the direct form of the hardware architecture adapting the original procedure, and a modified architecture for power reduction is proposed and discussed.

### *A. Direct Form of the System*

According to the density-based algorithm, we may derive the working schedule of the system directly as shown in Fig. 3. Two on-chip SRAMs are used to store the density information and the cluster label for each cell. The working schedule is briefly described as below. When the system detects the spike features, the system calculates the corresponding memory address, and adds proper kernel weights on the memory units instead of recording the raw features. In this design, a $3 \times 3$ symmetric kernel is applied, and one read-write procedure is required for 9 memory units when one set of the extracted features comes in. Therefore, in order to handle real-time density accumulation, the operation frequency should be faster than the rate of the input feature set. Next in the clustering state, the system iteratively retrieves the accumulated densities stored in SRAM 1 and writes the labeling results into SRAM 2. Finally in the merging step, the system allocates both the label and the density information stored on the two SRAMs to check the merging conditions. If there is a merging event, the cluster labels for some memory units may change, and the clustering results stored in SRAM 2 should be updated. After finishing the merging step, we can access SRAM 2 to get the final mapping table for the on-line processing device for classification.

### *B. Power Reduction: The Convolution Method*

Analyze the direct form of the hardware architecture, we can find that in each step during the classification procedure, the system needs to access the two SRAMs frequently for processing. Therefore, a large amount of memory access power may be consumed. We further observe that due to the large amount of collected features, about 80% of the memory access comes from the first step, that is, the on-line density accumulation. In order to reduce the considerable data access power, the concept of convolution is adapted. We first note that since the chosen profiling kernel is symmetric, the density of each cell is equal to the number of features within that cell convolutes the kernel. That is, for each cell $C_{x,y}$ in the feature space, the density $d_{x,y}$ can be computed as

$$d_{x,y} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} (k_{i,j}) \cdot (n_{x-i,y-j})$$

, where $k_{i,j}$ represents the kernel weights and $n_{x,y}$ is the total number of spikes detected within the cell $C_{x,y}$. $x$, $y$ are the corresponding coordinates of the cell in the 2D feature space. The procedure of convolution is also shown in Fig. 4 (a). Therefore, the first step for density accumulation can be separated into two stages. First, instead of accumulating the kernel weights on all the cells within the kernel when each feature comes in, we simply accumulates the number of features of the corresponding cell. Since only one memory unit needs to be accessed in this case, the operation frequency of the system can be slower than that of the direct form under the same input feature rate. In the second stage, we apply the convolution procedure to calculate the corresponding density for each cell sequentially. Note that during this procedure, a data reuse scheme shown in Fig. 4 (b) can be adopted. Through buffering the reusable data, the amount of data access can be further reduced.

With the modified two-stage procedure for density map construction, about 80% of the amount of memory access in the original on-line accumulating step can be saved. The total amount of memory access of the system can also be reduced to 40% of the original amount. Therefore, the reduction on power consumption should be available.

### *C. Memory Tradeoff for the Convolution Method*

Although the convolution method described above may achieve efficient power saving for the system, we should note that additional memory units are required to temporarily save the number of features during the procedure of convolution. Therefore, the area and the leakage power of the hardware may also increase. This problem is partially released by the scheme of memory reuse. As Fig. 5 shows, there should have been one more SRAM after inserting the convolution step which is used to store the feature numbers. However, we can find that the data life time of this SRAM will not overlap with SRAM 2 because the information of the feature number in each cell is used only during the convolution step. Therefore, we can reuse SRAM 2 for storing the number of features during the on-line feature accumulation, and the increment of memory usage caused by the convolution method can be minimized.

## V. Implementation Results

The training device for the classification part of the SSP with the proposed architecture is implemented in UMC 90nm

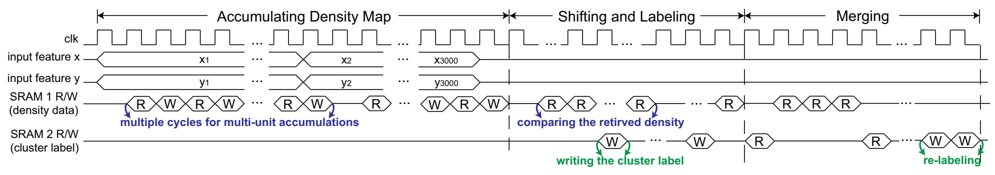
Accumulating Density Map
Shifting and Labeling
Merging
clk
input feature x
input feature y
SRAM 1 R/W (density data)
SRAM 2 R/W (cluster label)
x1
x2
x3000
y1
y2
y3000
multiple cycles for multi-unit accumulations
comparing the retirved density
writing the cluster label
re-labeling

Fig. 3. The working schedule of the direct form of the system. R: loading data from SRAM. W: writing data to SRAM.

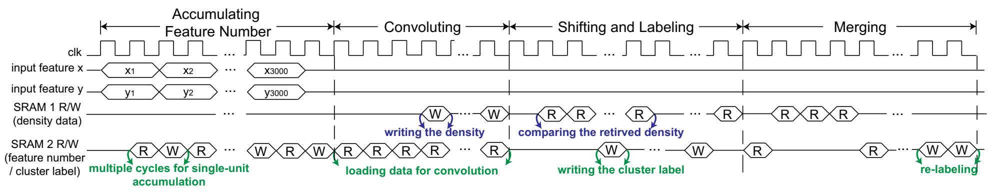
Accumulating Feature Number
Convoluting
Shifting and Labeling
Merging
clk
input feature x
input feature y
SRAM 1 R/W (density data)
SRAM 2 R/W (feature number / cluster label)
x1
x2
x3000
y1
y2
y3000
writing the density
comparing the retirved density
multiple cycles for single-unit accumulation
loading data for convolution
writing the cluster label
re-labeling

Fig. 5. The working schedule of the system that adapts the convolution method. R: loading data from SRAM. W: writing data to SRAM.

TABLE I
IMPLEMENTATION RESULTS IN 90 NM CMOS PROCESS

| | |
|---|---|
| Supply Voltage (Volt) | 1.08 |
| Operation Frequency (MHz) | 2 |
| Core Area (mm$^2$) | 0.25 |
| Power Consumption ($\mu$W) | 32.6 |

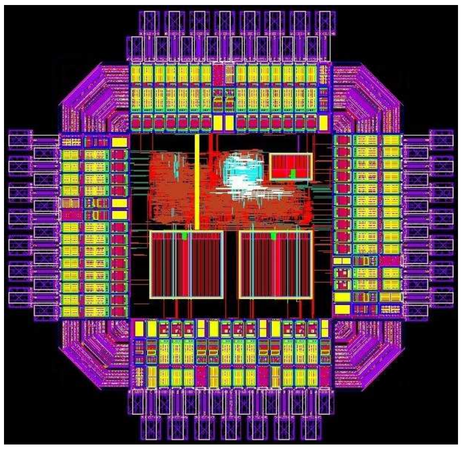

Fig. 6. The chip layout of the hardware implementation.

low-leakage CMOS process. Figure 6 shows the chip layout and table I summarizes the final implementation results. The operation frequency of the system is set to be 2MHz and is able to handle real time feature processing for 1M feature sets per second with a resolution of 16 bits per feature. About 3,000 spike feature sets can be recorded. The core area of the hardware is 0.25 mm$^2$ with 12.1k logic gates and 28.7kb SRAM. The average power consumption is 32.6 $\mu$W.

Figure 7 gives an estimated power comparison between different forms of the system based on the synthesis results. The input rate is set to be 1M feature sets per second for all cases. In order to maintain realtime performance, the operation frequency of the direct form is 18MHz. For other forms that adapt the convolution method, the operation frequency is

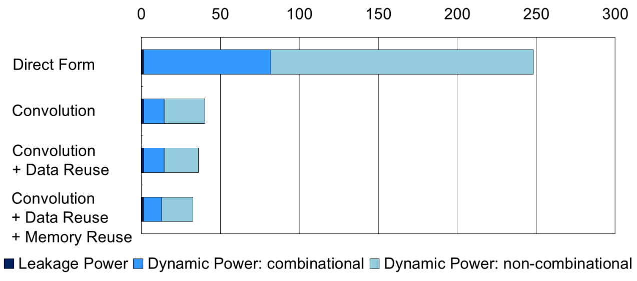

Fig. 7. The comparison of power consumption of different forms of the system.

2MHz. The dynamic power consumption for the combinational part and non-combinational part are estimated by the ratio of their area. We also assume that the dynamic power for the non-combinational part is proportional to the amount of memory access. According to the comparison results, the leakage power of the final architecture is slightly increased due to the 24% increment of memory usage. However, since the dynamic power is greatly reduced, the total amount of power consumption decreases. The final architecture efficiently saves about 87% of the power consumption.

## VI. CONCLUSION

In this paper, a training device of classification for the spike sorting processors is implemented. We first adapt the density-based hardware-oriented algorithm for the system because of the relatively small amount of memory usage and low computation complexity in comparison with the traditional k-means and mean shift algorithm. In the hardware architecture level, we involve the convolution method and the concept of data reuse to reduce the large amount of data access power when on-line accumulating the density map. The scheme of memory reuse is also adopted to minimized the increment of the required memory due to the convolution step. According to the final implementation results, the power consumption is 32.6 $\mu$W and the core area is 0.25 mm$^2$ in 90nm low-leakage process. This on-chip classification training device may be further integrated with real-time SSPs for the realization of embedded spike sorting microsystems.

## References

[1] M.D. Linderman and et al., "Signal processing challenges for neural prostheses," *IEEE Signal Proc. Magazine*, vol. 25, no. 1, pp. 18–28, 2008.

[2] Z. Zumsteg and et al., "Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems," *IEEE Tran. on Neural Syst. and Rehab. Eng.*, vol. 13, no. 3, pp. 272–279, 2005.

[3] M. Chae and et al., "A 128-channel 6mw wireless neural recording ic with on-the-fly spike sorting and uwb transmitter," in *ISSCC Dig. Tech. Papers*, Feb 2008, pp. 146–603.

[4] V. Karkare and et al., "A 130-gw, 64-channel spike-sorting dsp chip," in *ASSCC Dig. Tech. Papers*, 2009, pp. 289–292.

[5] T. C. Chen and et al., "A biomedical multiprocessor soc for closed-loop neuroprosthetic applications," in *ISSCC Dig. Tech. Papers*, Feb. 2009, vol. 25, pp. 434–435.

[6] L.-F. Cheng and et al., "Density-based hardware-oriented classification for spike sorting microsystems," in *Int. IEEE/EMBS Conf. on Neural Engineering*, 2011, pp. 170–173.

[7] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Comput. Neural Syst.*, pp. R53–R78, 1998.

[8] Q. Zhao and et al., "Evolving mean shift with adaptive bandwidth: A fast and noise robust approach," in *Asian Conf. on Computer Vision*, Sept. 2009, vol. 5994, pp. 258–268.