An Adaptive Neural Spike Processor With Embedded Active Learning for Improved Unsupervised Sorting Accuracy

Majid Zamani, *Member, IEEE*, Dai Jiang, *Member, IEEE*, and Andreas Demosthenous, *Fellow, IEEE*

***Abstract*—There is a need for integrated spike sorting processors in implantable devices with low power consumption that have improved accuracy. Learning the characteristics of the variable input neural signals and adapting the functionality of the sorting process can improve the accuracy. An adaptive spike sorting processor is presented accounting for the variation in the input signal noise characteristics and the variable difficulty in the selection of the spike characteristics, which significantly improves the accuracy. The adaptive spike processor was fabricated in 180-nm CMOS technology for proof of concept. It performs conditional detection, alignment, adaptive feature extraction and online clustering with sorting threshold self-tuning capability. The chip was tested under different input signal conditions to demonstrate its adaptation capability providing a median classification accuracy of 84.5% and consuming 148 µW from a 1.8 V supply voltage.**

***Index Terms*—Adaptive decomposition, brain machine interface, feature extraction, processor, reconfigurable embedded frames, signal model learning, spike sorting, unsupervised clustering.**

I.INTRODUCTION

INTERACTIONS between neurons are performed via electrical signals known as action potentials or spikes. The information of spikes from neurons has led to the development of miniaturized and implantable brain machine interfaces. These have been introduced for therapeutic applications using the neural modulation of a particular pathway [1], [2], as a communication bridge for control of assistive devices for patients with damaged sensory/motor functions (e.g. hand prosthesis [3], [4]) and restoration of lost cognitive function [5]. Such neural interfaces have benefited from advances in both electrode technology and microelectronics [6]-[8].

The detection of spikes by electrodes may involve the combined activity of typically 5 to 10 neurons [9]. Spike sorting is the process of grouping the recorded spikes into clusters based on the similarity of their shapes. As shown in Fig. 1, it comprises the following steps: 1) *detection and alignment*, separating spikes from noise and aligning the spikes to a common point, 2) *feature extraction*, extracting features of the spike shapes which gives a *dimensionality reduction*, i.e., going from a space of dimension N (with N the number of datapoints per spike) to a low dimensional space of a few features (K), and 3) *clustering*, grouping spikes with similar features into clusters (Z), corresponding to the different neurons. Spike sorting must account for the variety of spike shapes, different firing rates and noise [10]. This involves significant processing and therefore power which is a serious constraint in implant applications.

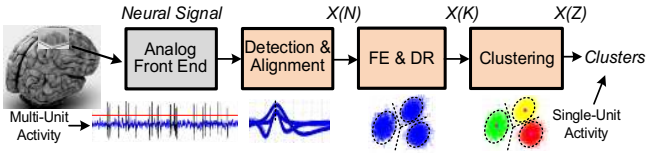

Fig. 1. Spike sorting process for determining single unit activity. Data is progressively reduced over processing blocks ($Z < K < N$).

Future spike sorting processors aim to improve accuracy and reduce power consumption suitable for implantable devices [11]. Integrated spike sorting processors have been developed [12]-[16]. In [12], the spike sorting processor performs detection, alignment and feature extraction but online clustering is not included. One of the most complete spike sorting processors is described in [13]. The design is multichannel, online and unsupervised and is compatible with the constraints of implantable devices. It achieves a median clustering accuracy of 75%. Note that the clustering accuracy decreases when there are close similarities between the recorded spikes [17]. A multichannel spike sorting processor that performs detection and feature extraction is described in [14]. Despite the use of a parallel-folding structure to reduce the hardware resources, it is not suited to implantable applications due to a power density which exceeds the safe limits for implants. An asynchronous spike sorting processor is described in [15]. The asynchronous self-timed methodology has inherent latency adjustment due to process variations but it provides a low power design. It consists of detection, alignment and feature extraction, but the clustering uses an external circuit. In [16] a real-time spike sorting processor is presented. It consists of spike detection, feature extraction and an improved clustering algorithm. The efficiency of this approach degrades with time due to the

Manuscript submitted November 20, 2017, revised, 27 February, 2018. This work was supported in part by a UCL PhD scholarship to M. Zamani.

M. Zamani, D. Jiang and A. Demosthenous are with the Department of Electronic and Electrical Engineering, University College London, Torrington Place, London WC1E 7JE, UK. (e-mail: m.zamani@ucl.ac.uk, d.jiang@ucl.ax.uk, a.demosthenous@ucl.ac.uk).

variation of noise and spike similarity. When the number of clusters is set manually its clustering accuracy is 87% and drops to 72% in online mode which is less than the reported median clustering accuracy in [13].

The common challenge in all the spike sorting processors in [12]-[16] is that they are not capable of adapting to the varying recorded neural signal characteristics such as background noise variations, electrode drift and appearance/disappearance of active neurons [18]. There is a need for a processor that adapts (learns) and embeds the high order signal models in the conventional spike sorting chain. In [19], the architecture and preliminary design of an adaptive spike sorting processor was introduced in which the signal model is captured through embedded frames and the processing chain is intermittently reconfigured to maintain optimal clustering performance.

This paper is a further development of [19]. The complete design, implementation and testing of the adaptive spike processor is presented, including confirmation of its successful adaptation providing high clustering accuracy. The remainder of this paper is organized as follows. Section II presents the general concept of adaptive spike sorting featuring embedded frames. Section III provides the architecture and system level details of an unsupervised, adaptive spike sorting processor for implantable applications. The measured results of the fabricated chip based on 180-nm CMOS technology are presented in Section IV. Section V concludes the paper.

II. Principle of Embedded Frames for Adaptive Spike Sorting

The general concept of embedding frames into a synchronous processor (SYNC DSP) to provide adaptive features is shown in Fig. 2(a). Intelligence (active learning) is incorporated into the SYNC DSP by embedding frames (Frame *a* … Frame *z*) which provide information about the captured model $h(x)$ of the input signal (x_1, x_2, … x_w). The frame information may be distributed to individual processing blocks of the SYNC DSP to allow dynamic adaptation.

Figs 2(b)-(e) show application of this concept to spike sorting. The two key factors in spike sorting performance degradation are the noise of the recorded data and the similarity index between the spike waveforms. The aim is to develop a spike processor in which the performance is automatically adjusted to an optimal level (maintaining lowest clustering error) accounting for different noise levels and the varying difficulty between the recorded spike waveforms. Fig. 2(b) is the block diagram of a conventional synchronous spike processor whose performance varies as a function of noise [*f*(Noise)] and similarity of extracted spikes [*f*(Similarity)]. Fig. 2(c) shows the spike sorting concept developed with added reverse-adjustment flow where the clustering performance (CA_{CC}) is independent (⊥) of noise and spike shape similarity. As shown in Fig. 2(d) this is captured in two frames (Frame 1 and Frame 2) for two variable parameters:

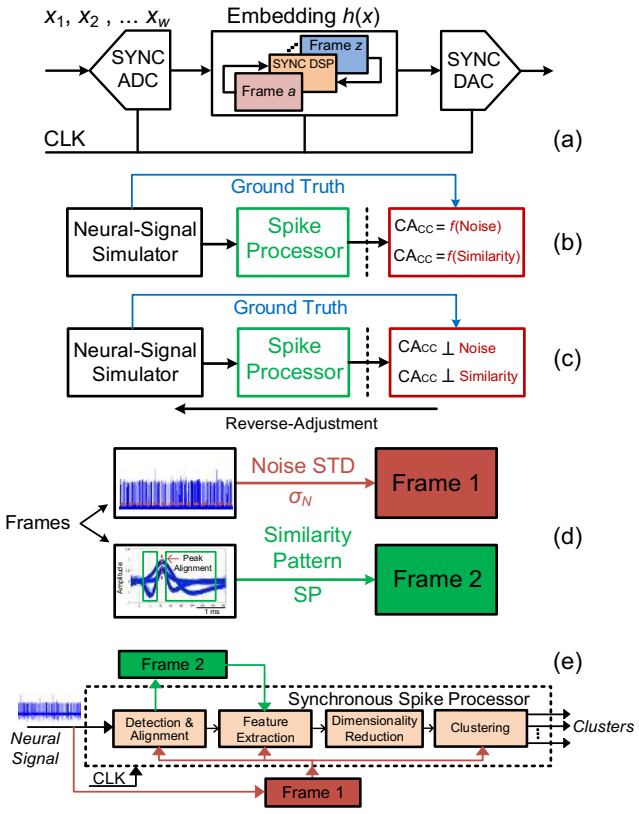

Fig. 2. (a) Conversion of a classical synchronous processor (SYNC DSP) to a processor with adaptive features. The characteristics of the captured model $h(x)$ of the input signal (x_1, x_2, … x_w) are embedded into the SYNC DSP using frames. (b) Conventional spike processor in which the clustering accuracy (CA_{CC}) is a function of input data noise [*f*(noise)] and spike similarity [*f*(similarity)]. (c) A spike processor independent (⊥) of input data noise and spike similarity. Here CA_{CC} remains approximately constant with varying noise and spike similarities. (d) Input noise standard deviation (σ_N) is captured in Frame 1 and the similarity pattern (SP) of the spikes in Frame 2. (e) Frame 1 is embedded into three blocks of the synchronous spike processor (detection and alignment, feature extraction and clustering) and Frame 2 is embedded into the feature extraction block only.

input noise standard deviation (σ_N) and the similarity pattern (SP) of the spikes. Adding the frames to the traditional spike processor presents a fundamentally new approach for mapping the recorded spikes to the individual neurons. Fig. 2(e) shows the two frames added to the spike sorting described in [20] to realize an adaptive spike processor. The adaptive processing provides an on-chip tuning mechanism for programming the key coefficients in the relevant building blocks. For Frame 1, σ_N can be evaluated by median processing of the recorded neural data. Frame 2 models the localized difference extraction of the aligned spikes as in [21]. SP is intermittently updated with the similarity information of the latest spike waveforms.

III. Adaptive Spike Sorting Processor

A. System Architecture

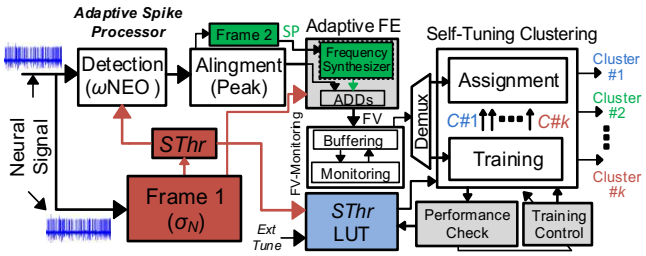

Fig. 3. Adaptive spike sorting architecture. The frames in Fig. 2 are embedded into the spike processor (Frame1 = σ_N; Frame 2 = SP). It is designed to distinguish up to six (k = 6) active neurons in the recording channel.

Fig. 3 shows the architecture of the adaptive spike processor using the embedded frames. The amplified, band-pass filtered and digitized neural data is sent to the adaptive spike processor. Frame 1 monitors the noise standard deviation (σ_N) of the neural data and defines the sorting threshold $SThr = 4 \cdot \sigma_N$ [22] which is distributed to the detection block, adaptive feature extraction (FE) block and sorting threshold look-up-table (*SThr* LUT). The SP extracted in Frame 2 is sent to the frequency synthesizer (FS) for tuning the decomposition lines. Each spike is extracted using a 2.5 ms window and aligned to a common temporal reference. In the detection block, the *SThr* is considered as a conditional activation function of the modified version of a nonlinear energy operator (ωNEO) [23]. The authenticity of spikes are examined with *SThr* and ωNEO. The spike detection power used is significantly reduced by masking the worthless data and inhibiting the asynchronous initiation of the detection block.

In the adaptive feature extraction block, extrema sampling [20] of adaptive discrete derivatives (ADDs) provides an efficient method not only in computational simplicity but also in accuracy to transform the recorded spikes to a feature space that better separates the different neurons. Selective spike decomposition is performed using the aligned spike waveforms and FS. SP is updated over time to monitor the similarity level between the extracted and peak-aligned spikes. Feature extraction is adjusted to the appropriate sub-bands (decomposition sub-bands) with the most informative samples based on the FS output. The maximum separation between the spikes is achieved by extrema sampling of selected sub-bands. The feature vectors (FVs) are sent to the features monitoring block (FV-monitoring) and subsequently to the clustering block.

The modified version of the online sorting algorithm (O-Sort) in [24] is used for real-time and unsupervised clustering of neurons. The cluster means identified in the training phase (*C*#1 … *C*#*k*) are saved in the memory of the assignment block. During the cluster-mapping phase, the input FVs are mapped based on their minimum distance to one of the identified cluster means saved during the training phase. The performance check and training control blocks are exploited in the clustering block to enhance the clustering median accuracy by incorporating a sorting threshold self-tuning scheme. The performance check block monitors and evaluates the clustered FVs based on the defined performance metrics. It decides

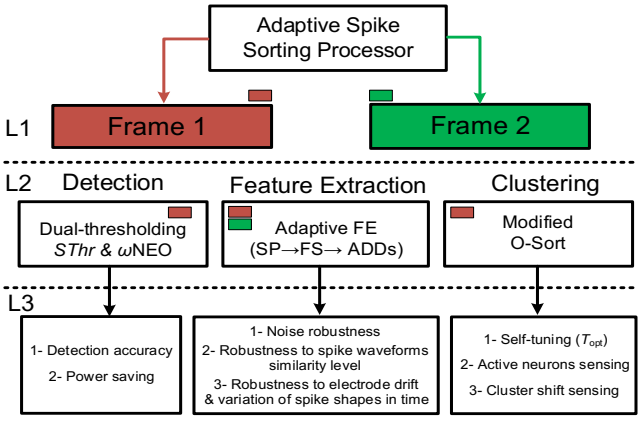

Fig. 4. Adaptive processing hierarchy. Robustness is increased via the red path for noise and via the green path for the variation in the spike waveforms. Layer 1 (L1): sensing frames; Layer 2 (L2): allocation of frame functions for detection, feature extraction and clustering; Layer 3 (L3): features associated with each allocation.

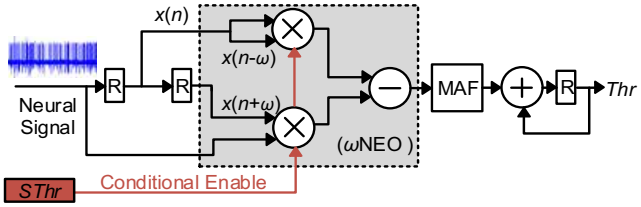

Fig. 5. Implementation of ωNEO with conditional control. Conditional enable is initiated by $SThr = 4 \cdot \sigma_N$. The moving-average filter (MAF) reduces the noise effect from the ωNEO output signal and improves the accuracy of the calculation of the detection threshold *Thr*. ω = 2 in this design.

whether the level of the sorting threshold should be iteratively adjusted to an optimal level (T_{opt}) [20] in the *SThr* LUT block, and if needed triggers retraining to re-compute the cluster means.

Fig. 4 shows the hierarchy of the functions providing the derived features from the adaptive spike processor. In the following sections further details of the operational aspects are described.

B. Detection and Alignment

The nonlinear energy operator (NEO) [23] is an unsupervised method for calculating the energy variation of the original signal to interpret the spike events in time. NEO is defined as:

$$\psi(n) = x^2(n) - x(n+1) \cdot x(n-1) \qquad (1)$$

where $x(n)$ is the input digitized signal and $\psi(n)$ is the NEO value at sampling point n. This operator highlights the large variations in power and frequency. The characteristic of spike activity is instantaneous. The NEO operator emphasizes the amplitude-energy variation of the spikes and improves the signal to noise ratio (SNR) in a noisy environment.

However, NEO is poor in the detection of spikes with low frequency components. To increase robustness to spike amplitude variations and reduce out-of-band noise sensitivity, (1) is changed to $\psi(n) = x^2(n) - x(n+\omega) \cdot x(n-\omega)$,

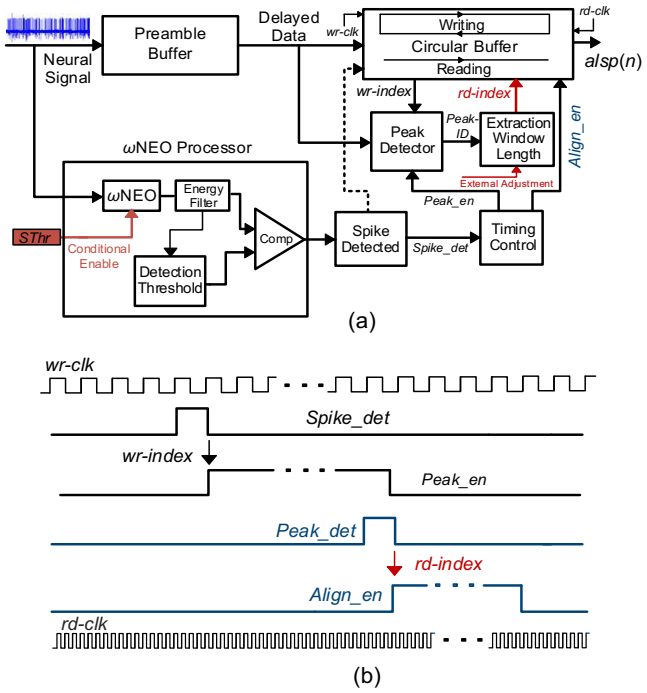

Fig. 6. (a) Architecture of the detection and alignment block. (b) Timing diagram.

where ω is between 1 and 3 by experiment. This is defined as ωNEO.

Fig. 5 shows the block diagram of the ωNEO conditional control function. This approach has two advantages. Firstly, conditional enabling is directly applied to the ωNEO block, thus when the input exceeds the clustering threshold (*SThr*), dual-thresholding (*SThr* and ωNEO) is executed providing a double check on accuracy. Secondly, dual-thresholding provides a power reduction of ~30% (based on Cadence synthesis simulations).

The conventional method used for threshold calculation at the output of ωNEO is energy accumulation divided by the window sample numbers. The power variations in different simulations show that the output of the ωNEO is sensitive to noise disturbances. Normally the input signal to the ωNEO is composed of spike events which exhibit localized energy of a specific duration and other samples as a result of noise interference. The output of ωNEO can be sensitive to breakthrough of noise from the input signal. To minimize this effect a simple moving average filter (MAF) is applied in Fig. 5. The detection threshold *Thr* is calculated as:

$$Thr = \frac{\alpha}{N}\sum_{n=1}^{N}\lambda(n) \qquad (2)$$

where $\lambda(n)$ is the filtered signal energy, N is the number of samples per window, and α is a constant (empirically chosen to be 8 in this implementation). To reduce the buffering of the threshold calculation, the detection threshold is updated per window rather than per sample. The calculated threshold is used for the next segment of data; the accumulator is reset and starts again for the next data window.

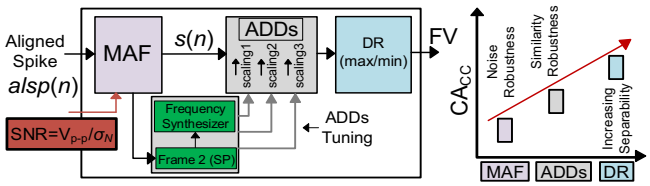

Fig. 7. Adaptive feature extraction architecture. The MAF suppresses the effect of random high frequency noise of the aligned spike waveform *alsp*(n). The Frame 2, frequency synthesizer and ADDs provide adaptive decomposition. Three decomposition lines are selected based on scaling1, scaling2 and scaling3. The dimensionality reduction (DR) block uses extrema (max/min) sampling of the decomposed spike waveform *s*(n).

Fig. 6 shows the detection and alignment block architecture. The input neural data is sent to both a preamble buffer and the ωNEO block. The preamble buffer is a digital delay line of 24 cells. It synchronizes the ωNEO output with the starting point of a spike and buffers the samples before the spike exceeds the threshold level. The delayed data is continuously written to a circular buffer. When a spike is detected, the corresponding writing index (wr-index) is sent to the peak detector block and thus the sample counting and peak address are synchronized. The output of the peak detector (peak-ID) is used to define the extraction window length. The peak-ID is the fifteenth sample of the 45 samples in the aligned window. The read index (rd-index) representing the first sample in the window, is sent to the reading block of the circular buffer and the aligned spike samples *alsp*(n) are transferred to the adaptive feature extraction block. The reading clock rate is 4x faster than the writing clock rate to ensure capturing spikes that are close in time.

C. Adaptive Feature Extraction

Feature extraction transforms the aligned spikes to a low-dimensional space and emphasizes the spike waveform differences. Fig. 7 shows the adaptive feature extraction block, a modified version of [20]. It consists of a MAF, frequency synthesizer (FS), adaptive discrete derivatives (ADDs) and dimensionality reduction (DR) blocks. The MAF acts as a denoising filter to improve feature extraction robustness to random noise (out-of-band noise) while retaining the crucial encoded information buried in the spikes. The SNR = $V_{\text{p-p}}/\sigma_N$ obtained from Frame 1 is used to decrease noise sensitivity and increase feature extraction separability by adjusting the length of MAF.

MAF averages a specific number of samples of the incoming aligned spikes *alsp*(n) to produce the smoothed output signal *s*(n) expressed as:

$$s(n) = \frac{1}{M}\sum_{j=0}^{M-1} alsp(n-j) \qquad (3)$$

where M is the filter length. M is defined based on SNR for different σ_N: $M|_{\text{SNR}=V_{\text{p-p}}/\sigma_N=0.1,0.15,0.2} = 1,2,4$.

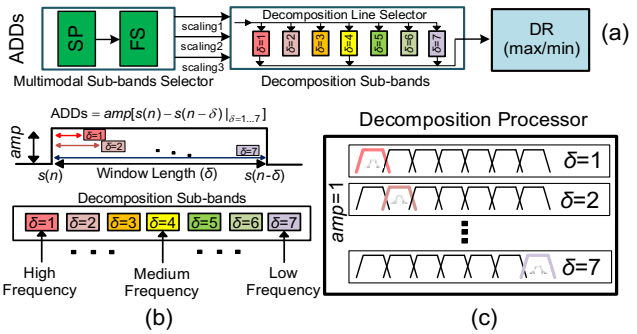

Fig. 8. (a) Illustration of sub-band selection using the sensing path SP→FS. Three sub-bands that accommodate the most separable feature are chosen by scaling1, scaling2 and scaling3. (b) Feature extraction employing spectral analysis in the ADDs. Decomposition intensity range is shown with different colors from high (δ=1) to low (δ=7). (c) Illustration of ADDs as an adaptive filter.

The ADDs block calculates the slope at each sample point over a number of different time scales:

$$\text{ADDs} = amp[s(n) - s(n-\delta)|_{\delta=1\ldots7}] \qquad (4)$$

where *amp* is the amplitude of the decomposition window (here set to 1), *s* is the spike waveform, *n* is the sample point and δ is the scaling factor (time delay). Adjustment of the scaling factors (scaling1, scaling2, scaling3) is based on three frequency sub-bands from $\delta = 1$ to $\delta = 7$ corresponding to the most informative features (non-Gaussian features) for clustering as shown in Fig. 8. A sensing path (SP → FS) monitors the localized differences between the spike waveforms and distinguishes the three informative sub-bands for tuning in the decomposition processor (SP → FS→ ADDs). The sensing path inserts robustness to high degrees of similarity in the spikes. Placing the sensing chain before the decomposition processor reduces the hardware resources and improves the effectiveness of spike waveform disintegration. The path SP → FS → ADDs can be implemented by considering all parallel decomposition sub-bands from $\delta = 1$ to $\delta = 7$ and applying multimodal metric in each decomposition line to retain the separable features.

The frequency synthesizer (FS) converts the extracted localized differences pattern to the sub-bands with the most informative parameters for clustering. The frequency synthesizer operation is shown in Fig. 9. Having generated SP as shown in Fig. 9(b), analysis of its slope variations is performed to assign weights to the range of variations from high ($\delta = 1$) to low ($\delta = 7$). The slope uses the first derivative:

$$\text{FD}_{(\text{SP})} = \text{SP}(n) - \text{SP}(n-1) \qquad (5)$$

The frequency variation range (FVR) of SP is defined as:

$$\text{FVR} = \left|\text{FD}_{(\text{SP,max})} - \text{FD}_{(\text{SP,min})}\right| \qquad (6)$$

where $\text{FD}_{(\text{SP,max})}$ and $\text{FD}_{(\text{SP,min})}$ are the maximum and minimum of the $\text{FD}_{(\text{SP})}$. FVR is divided into seven scales to cover all the possible frequency range [high (A) to low (G)] as shown in Fig. 9(c)-(d). The absolute value of $|\text{FD}_{(\text{SP})}|$ is then synthesized into the defined frequency ranges. Once the weight allocation

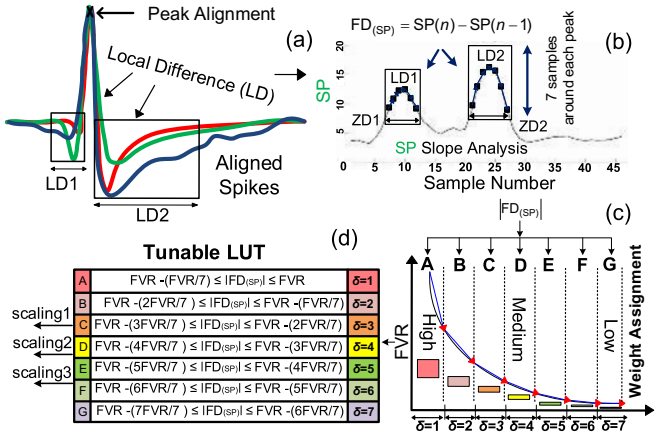

Fig. 9. Frequency synthesizer (FS) and scaling factor (δ) tuning. The process is illustrated using the C_Difficult1 dataset [26]. (a) Peak aligned mean waveforms in C_Difficult1. Local differences are identified in LD1 and LD2. (b) Generated similarity pattern (SP = Frame 2) from the accumulated local differences. ZD1 and ZD2 identify the zero differences. The first derivative of the local differences (seven samples around each peak) provides the active frequencies in SP. (c) Weight assignment to the absolute value of $|\text{FD}_{(\text{SP})}|$ corresponding to the sample numbers in LD1 and LD2. The decomposition intensity is shown in different colors from high (δ=1) to low (δ=7). The winner sub-bands represented by scaling1, scaling2 and scaling3 are used to tune the decomposition processor.

process has been performed, the three scaling factors (scaling1, scaling2 and scaling3) with the highest weights are chosen for tuning the ADDs.

The proposed approach for adaptive decomposition of spike waveforms is similar in operation to the methods in [20] and [22]. The feature extraction method in [22] employs four-level multi-resolution decomposition using Haar wavelets which result in 64 wavelet coefficients for each spike. Then the Kolmogorov–Smirnov (K-S) test [25] for normality is applied to select the first 10 informative features in the examined datasets [26] as shown in Fig. 10(a). The combination of Haar wavelets and K-S test is developed for offline processing and it requires large amount of hardware resources. The feature extraction method in [20] [Fig. 10(b)] uses discrete derivatives and extrema sampling for on-chip hardware realization purposes.

In the new proposed method for informative decomposition, Haar wavelets are replaced by parameterized ADDs and the K-S test is replaced by the sensing path SP → FS which is simply tuned over time. Different combinations are introduced in [20] by sweeping the decomposition window length (δ) to explore the frequency sub-bands (from $\delta = 1$ to $\delta = 7$) which accommodate the most informative features for the examined datasets [26]. By applying the multimodality metric it maintains the features exhibiting multiple peaks and valleys in their distributions. In [20], the process of choosing the combination with the highest clustering accuracy is performed offline. It is replaced here with the online and tunable informative sub-bands selector (SP →FS → ADDs). The hardware implementation of ADDs is shown in Fig. 10(c). It comprises adjustable delay lines, subtractors and dimensionality reduction blocks. They perform extrema

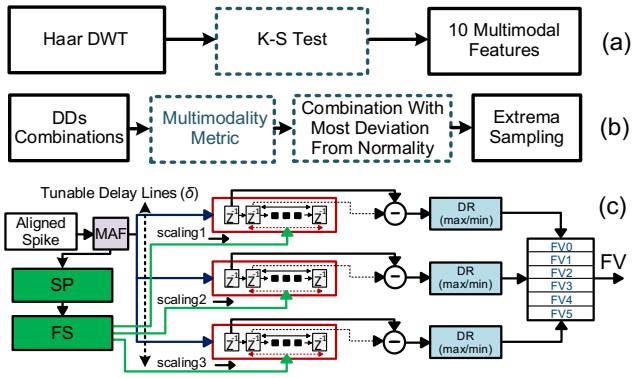

Fig. 10. Feature extraction methods in [22] (a), and [20] (b). (c) Block diagram of adaptive feature extractor. Three decomposition lines are activated and six features are selected for clustering. In this feature extractor, the multimodality metric is moved before the decomposition block to achieve high performance via selection of the decomposition sub-bands with multimodal features while keeping the complexity low.

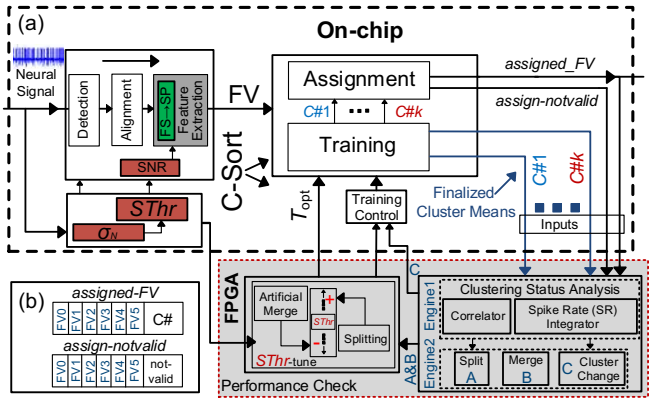

Fig. 11. Implementation of configurable online sorting (C-Sort). C-Sort enhances clustering performance robustness with little energy and hardware overhead. The blocks highlighted in the grey area determine the optimal sorting threshold (T_{opt}). The C-Sort is an "error-aware model" since it adapts the noise level and iteratively tunes it to an optimal value by undoing the effects of non-idealities in feature space.

selection of the decomposition lines to create a feature vector (FV). The proposed feature extraction is flexible in terms of frequency band selection and extraction of a wide range of features. These processes result in robustness to spike similarity and noise level in the feature extraction operation.

D. Clustering

Clustering provides classification of spikes into different groups, corresponding to different neurons. The clustering algorithm (O-Sort) in [24] is well suited for real-time neuron mapping. However, for cases where neurons are hardly distinguishable or there is significant background noise, the clustering accuracy in [24] when the sorting threshold (*SThr*) is at a non-optimum level, is severely degraded. This results in cluster splitting and artificial clustering causing errors in classification. To boost the clustering accuracy configurable online sorting (C-Sort) is proposed here (see Fig. 11) by including adaptive tuning of *SThr* to an optimal level (T_{opt}). The principles used include active embedded sensing of noise variations (Frame 1) for clustering error tolerance enhancement which defines *SThr*, and the detection of the clustered feature space non-idealities, for example, cluster split (Engine2-A in Fig. 11) to achieve error-aware clustering. To identify the number of active neurons in the recorded data, C-Sort includes a cluster change block (Engine2-C in Fig. 11). The added functions boost the clustering reliability providing resilience against statistical errors with little overhead in terms of power requirements and hardware.

1) Clustering State Machine and Operation:

A single-channel clustering function is shown in Fig. 12. It is divided into five time intervals: hold (t_0), training (t_1), validation (t_2), assignment (t_3) and cluster change ($t > t_3$). The clustering execution begins with t_0. The embedding frames (Frame 1 = σ_N; Frame 2 = SP) are initiated to calculate the signal characteristics and other parameters such as *SThr*. Training begins at t_1 when the initial value of *SThr* is identified. *SThr* is sent to the sorting *SThr* LUT (see Fig. 3) as

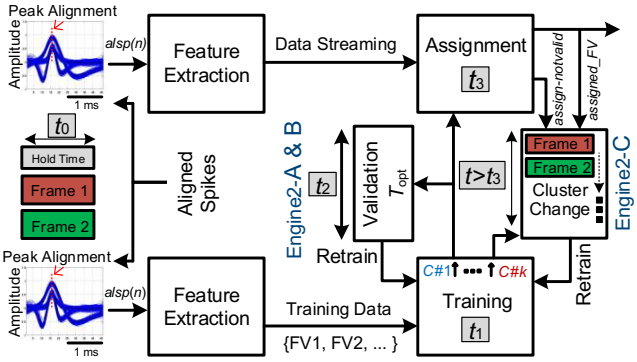

Fig. 12. Illustration of multi-phase clustering.

the initial value for training. The training period is tunable and is defined based on the number of feature vectors (FV1, FV2 …) to identify the cluster means in the recording channel. After training to evaluate the mapped data to the converged cluster means, t_2 is initiated. During this time interval, the performance check block (grey section in Fig. 11) is used to distinguish the clustered feature space non-idealities (e.g. cluster split) and adaptively fine-tune *SThr* to an optimal level (T_{opt}). Once the validation is performed (a maximum of three iterations) the identified cluster means (*C*#1 … *C*#*k*) are transferred to the assignment block as shown in Fig. 11. At t_3 the recorded spikes are continuously mapped to their origins. At $t > t_3$ Frame 1 and Frame 2 are updated intermittently to project either the trajectory movements of the existing active neurons in feature space (due to the noise or spike template amplitude fluctuations) or to reflect the appearance/disappearance of active neurons in feature space. After training the frames, the performance check block tracks and evaluates the updated feature space projection to decide whether or not channel re-training is required.

2) Performance Check:

The performance check block (see Fig. 11) identifies T_{opt} in

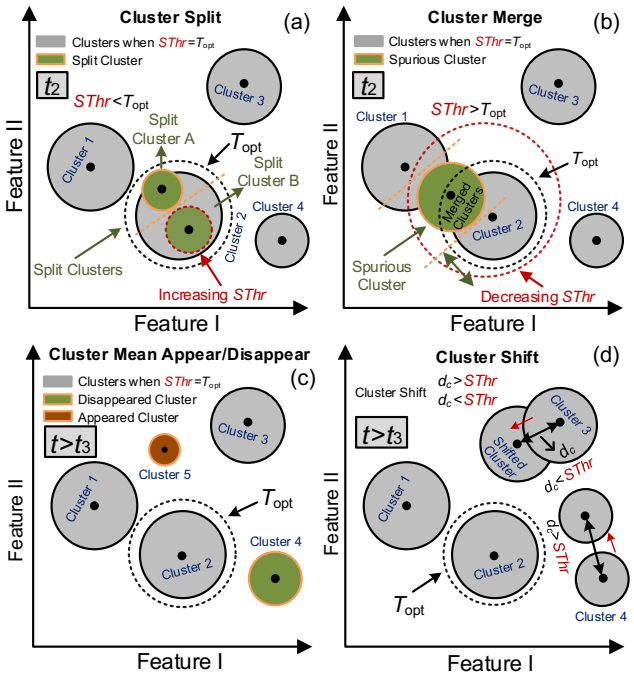

Fig. 13. 2-D illustration of cluster feature space for (a) cluster split, (b) cluster merge, (c) cluster disappearance/appearance and (d) cluster shift due to the noise or spike template amplitude fluctuations. If $d_c < STh$ the cluster mean (Cluster 3) is shifted to a new position in feature space; if $d_c > STh$ the case of appearance of a new cluster (Cluster 4) is valid. d_c is the distance between the cluster means (centroids).

t_2 and cluster change analysis in $t > t_3$. The inputs to the clustering status analysis block are the finalized cluster means ($C\#1$ … $C\#k$), assigned-FV and assign-notvalid as shown in Fig. 11(a). The structure of assigned-FV and assign-notvalid are shown in Fig. 11(b); assigned-FV comprises FV and the cluster number ($C\#$) while assign-notvalid comprises FV and not-valid flag to monitor the assignment error rate (not-valid flag is set when FV is not matched with any converged clusters in t_1). Engine1 comprises a correlator and a spike rate (SR) integrator. Its output provides the state of the clustered feature space to trigger Engine2. The latter has cluster split (A), cluster merge (B) and cluster change (C) blocks. Their operations are summarized as follows:

- *Artificial splitting* (Engine2-A): In the case of artificial splitting into multiple clusters [Fig. 13(a)], the correlation between the split clusters identified in the cluster means ($C\#1$ … $C\#k$) is high (> 0.9) and their SR is less than other active neurons. *STh* is increased to establish a hyperplane that forms an optimal clustering boundary between the existing clusters.
- *Artificial clustering* (Engine2-B): In the case of artificial clustering [Fig. 13(b)], a spurious cluster is created. *STh* is decreased for corrected clustering.
- *Variations in recorded neural data* (Engine2-C): When there is appearance/disappearance of active neurons [Fig. 13(c)] or cluster shift [Fig. 13(d)] as a result of noise or spike amplitude variation over time (e.g. due to electrode drift), the performance check block (Fig. 12) detects them. It then reinitializes training for detection of changes in the recorded neural data.

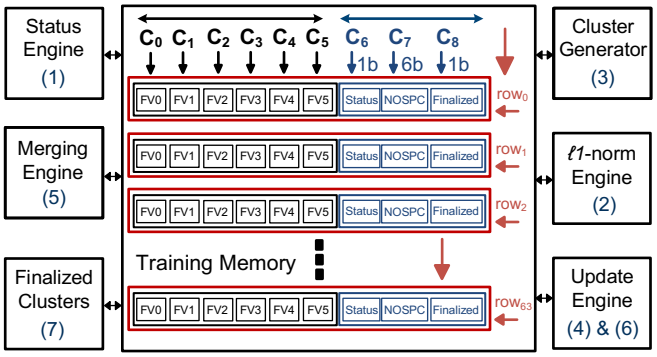

Fig. 14. Structure of the training unit (see Fig. 12) which comprises a training memory and peripheral processing engines (1-7); see also Fig. 15. Each row of training memory consists of six columns (C_0-C_5) for accommodating the extracted feature vectors (FV0-FV5), a 1 bit status flag (C_6) for dynamic power saving, 6 bits (C_7) for the number of spikes per cluster (NOSPC) for cluster mean update in (4) and (6) (C_7 is also used for cluster generation and checking the finalized cluster means in the training phase), and a 1 bit finalized flag (C_8) for conditional initiation of (4) and (6). The number of interleaved processing in $\ell 1$-norm (2) and merging (5) engines is chosen 8 to minimize the power-area product.

Adjustment of *STh* is performed over three different runs where the value of *STh* is modified from the initial value by 10% in each run ($\Delta = \pm 0.1$). This provides an improvement in median clustering performance of 5-8%. The performance check block also tracks changes in the number of active neurons and cluster shifts.

3) Training Unit Structure:

The training memory and the main processing engines are shown in Fig. 14. The flowchart of the operations performed by the engines is summarized in Fig. 15 using the adapted O-Sort algorithm. The training memory in Fig. 14 is implemented in a matrix format to provide highly flexible access to the memory locations. Status engine screens the activity of training and monitors the duration of training. When a FV is sent to the training block, it is compared to the existing transient cluster means in the $\ell 1$-norm engine whose block diagram is shown in Fig. 16. The minimum distance $d_{\min}$ between the FV(n) and the created transient cluster $c_i(n)$ is computed using the $\ell 1$-norm metric:

$$d_{\min} = \arg_{\min} \sum_i \sum_{n=1}^{N_S} |\mathrm{FV}(n) - c_i(n)| \qquad (7)$$

where N_S is the number of features and i (= 0, …, 63) is the number of rows in the training memory as shown in Fig. 14. If $d_{\min} < STh$, the FV is assigned to the existing cluster and the cluster mean is updated to be the weighted average of the first two spikes, otherwise a new cluster is automatically created and FV is assigned to it. When $d_{\min} > STh$, cluster creation engine provides an ID for a new transient cluster and if $d_{\min} < STh$ the on-hold FV is used for cluster mean update C_{update}:

$$C_{\text{update}} = \frac{W \cdot c(n) + \mathrm{FV}(n)}{W+1} \qquad (8)$$

where W is the number of spikes in a specific cluster (NOSPC-C_7).

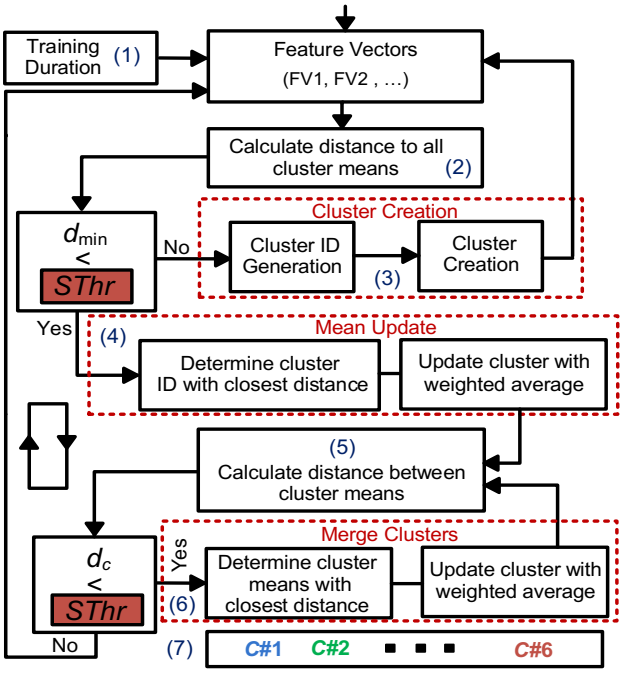

Fig. 15. Flowchart of unsupervised clustering algorithm C-Sort. Operation functions are annotated (1-7) based on the training engines in Fig. 14. The procedure is iterative.

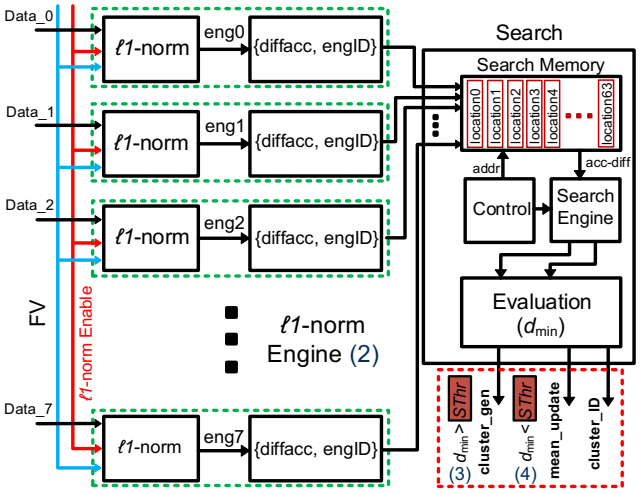

Fig. 16. Details of the $\ell 1$-norm engine. It consists of eight parallel processors and it is reused eight times to calculate the $\ell 1$-norm difference accumulation for all the 64 rows (row_0-row_{63}) as shown in Fig. 14. When a FV (blue line) is sent to the training block (see Fig. 11), initially it is compared to the transient cluster means created in the previous phases (Data_0 to Data_7). The minimum update distance (d_{min}) is calculated sequentially on the transient values in the search block to either activate the cluster generation (cluster_gen) or to revise the mean update value (mean_update).

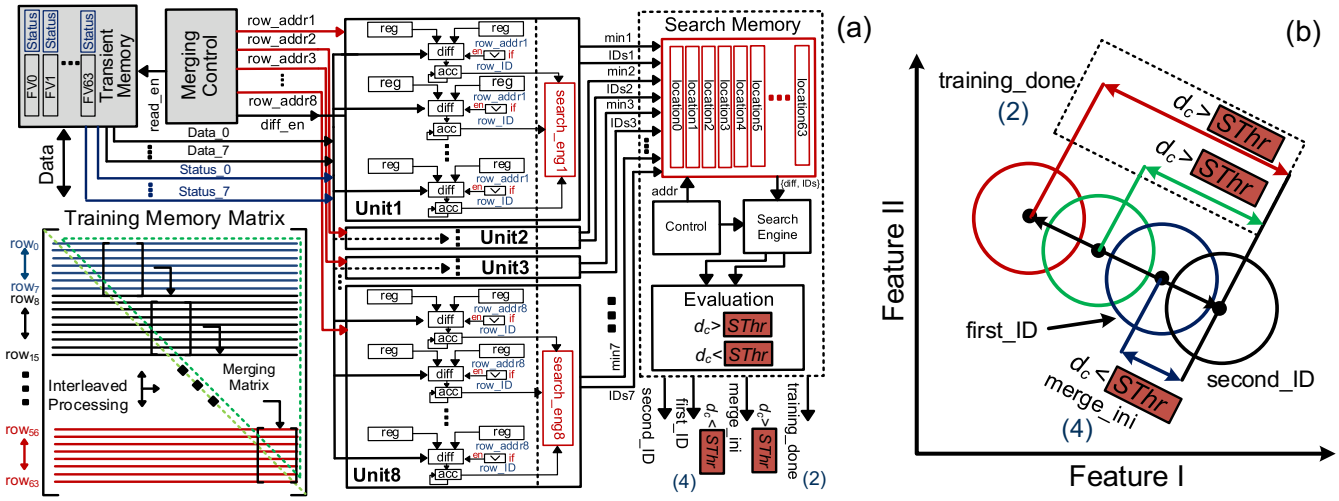

Fig. 17. (a) Details of merging engine (5). Two clusters with $d_c < STHr$ are indistinguishable and are merged. Interleaving of 8 parallel merging-check units is used to access the 64 matrix memory rows. At the end of the merging process, either merging update (merge_ini) is initiated or training is terminated for the current FV (training_done). In the case of merging update, the appropriate content of the memory rows based on the chosen IDs (first_ID and second_ID) are accessed. (b) 2-D illustration of distance between the cluster centroids.

Due to the cluster shift, there might be overlap between the clusters. In this case, two clusters with distance between their means (centroids) of $d_c < STHr$ in the feature space are indistinguishable and they are merged. To evaluate the merging possibility, the distance between all cluster means are calculated in the merging engine as shown in Fig. 17 and the selected candidates (first_ID and second_ID) are sent to the update engine. The centroid of the new merged cluster is calculated as a weighted mean:

$$C_{\text{merge\_update}} = \frac{W_1 \cdot c_1(n) + W_2 \cdot c_2(n)}{W_1 + W_2} \quad (9)$$

where c_1 and c_2 are the centroids, and W_1 and W_2 are the respective spike populations of each cluster. $C_{\text{merge\_update}}$ is stored in one memory location and the content of other locations is erased to be reused for subsequent cluster generation.

To reduce area-power circuit techniques such as interleaving, logic reusing and transient memory allocation

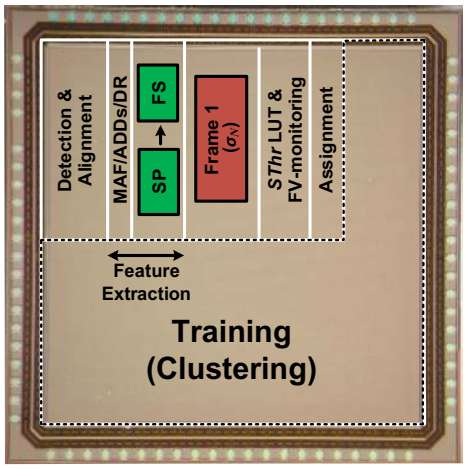

Fig. 18. Die photo of the adaptive spike sorting processor chip with processing blocks identified. The area is dominated by the training function for clustering. In a multi-channel processor, the training block would be shared between the recording channels.

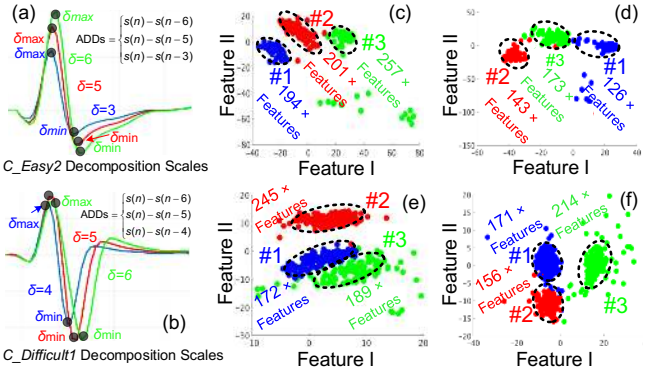

Fig. 19. Illustration of extrema features using different sets of scaling factors. The scaling factors are selected based on the frequency synthesizer output for (a) C_*Easy2_0.05* and (b) *C_Difficult1_0.05*. 2-D projection of clusters for (c) *C_Easy1_0.05*, (d) *C_Easy2_0.05,* (e) *C_Difficult1_0.05* and (f) *C_Difficult2_0.05*. (Spikes have been colored according to the ground truth). The number of features allocated to each cluster in the assignment phase takes 7 seconds. The 2-D projection plots are based on the most informative features (Feature I and Feature II).

TABLE I
ADAPTIVE SPIKE PROCESSOR PERFORMANCE AND FEATURES

Chip Summary	
Technology	180 nm
Supply Voltage	1.8 V
Power Consumption	148 µW
Core Size	6 mm^2
Clock Frequencies	30, 120, 240, 960 kHz
Processor Type	Adaptive
Median P_D, P_{FA}	92%, 1%
Median CA$_{CC}$	84.5%
Compression Factor*	150X**, 240X
Memory Size	10 kB
Input Data Rate	240 kbps
Active Learning Features	
Training Model	Embedded Frames
Function of Frames	Frame 1 = Noise Standard Deviation (σ_N) Frame 2 = Similarly Pattern (SP)
Frame Training Interval	Every 1 min**
Active Tuning Features	
Detection	*SThr* & ωNEO
Feature Extraction	SNR → MAF Length (M) SP → FS → ADDs
Clustering	C-Sort

* Ratio between the spike processor input data rate and its output data rate.
** Compression factor in error monitoring mode.
** The training time can also be tuned externally.

reusing were used in the training block of the clustering (see Figs 14, 16, 17).

IV. CHIP MEASURED RESULTS

The adaptive spike sorting processor was fabricated in a 180-nm CMOS technology for proof-of-concept. The die micrograph is shown in Fig. 18. The chip core area [1] occupies 6 mm^2. The processor uses four different clock rates (30 kHz, 120 kHz, 240 kHz, 960 kHz) to obtain the best processing efficiency and consumes 148 µW from a 1.8 V supply voltage. To evaluate the spike detection performance the following metrics are used: 1) probability of detection, P_D = TDS/TNS where TDS is the number of truly detected spikes and TNS is the total number of spikes; 2) probability of false alarm, P_{FA} = FD/TDS where FD is the number of false detections and TDS are the true positives. Table I summarizes the features and performance of the adaptive spike processor chip.

In the following sections, various testing methodologies are used to evaluate the chip performance under different conditions including confirmation of its successful adaptation providing high clustering accuracy.

A. Static Test

The static test examines the processor performance different spike shapes and different noise levels with a known ground truth. The spike datasets in [26] (Easy1, Easy2, Difficult1 and Difficult2) were used. Each dataset has three different types of spike shape and four different noise levels with standard deviations of 0.05, 0.1, 0.015 and 0.2 (each dataset contains 1.44 million samples). Fig. 19(a)-(b) shows cases for different scaling factors (δ) used for decomposition of spike waveforms. Scaling factors in the ADDs provides enhanced clustering discrimination. Extrema sampling provides six features for clustering. Fig. 19(c)-(f) shows the two-dimensional (2-D) projection of the clusters in all datasets. The boundaries of the clusters are identified by dotted lines. An overall median clustering accuracy of 84.5% is achieved.

B. Dynamic Test

A dynamic test to evaluate the adaptivity of the processor was used. To simulate dynamic variations in the data over time a random data selection procedure was used. The neural simulator employed the 4 standard datasets (*a*: *Easy1*, *b*: *Easy2*, *c*: *Difficult1*, *d*: *Difficult2*) each with its 4 noise standard deviations (*a'*: 0.05, *b'*: 0.1, *c'*: 0.015, *d'*: 0.2) - i.e. 16 different combinations. Five minutes of data was chosen

[1] It comprises 2.7 mm^2 non-training area and 3.3 mm^2 training area. The logic cells occupy 55% of the core area and the rest is for routing (only 4 metal layers are available in the 180-nm CMOS technology used). If the design was implemented in a deep sub-micron technology, e.g. TSMC 65-nm (9 metal layers), the logic area would scale down to 0.44 mm^2 (the area scaling factor form 180-nm to 65-nm is 7.67 [27]) and the routing area would be also much reduced.

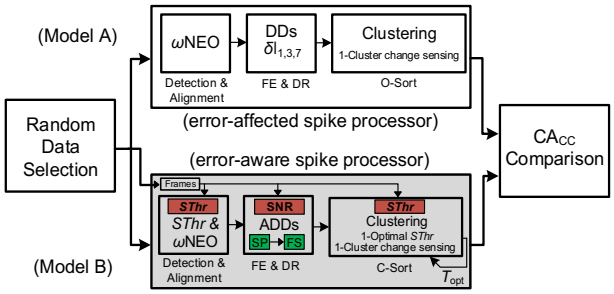

Fig. 20. Set-up for comparing the performance between model A (error-affected) and model B (error-aware) spike processors. The error-aware model employs adaptation of the input signal model enabled by the embedded sensing frames.

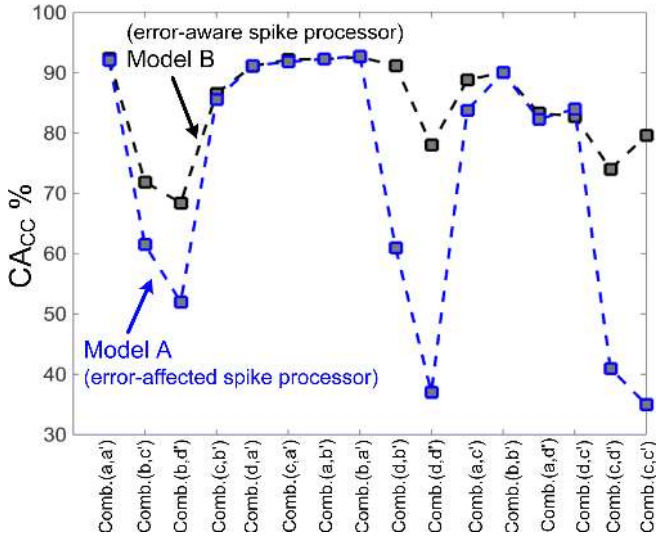

Fig. 21. Clustering accuracy (CA_{CC}) comparison between the model A (error-affected) and model B (error-aware) spike processors. The input data to the processors was randomly selected. For example, Comb.(a, a') is Easy1 with the lowest noise level (0.05). Model B is less affected by the dynamic changes in the input data.

randomly from each of the combinations and concatenated into a continuous stream of data providing variable data conditions over time.

Fig. 20 shows the alternative models of operation of the spike processor. In model A (error-affected model), the spike processor was configured to operate without the embedded frames. The constituent building blocks in this model are NEO based detection, multi-resolution decomposition utilizing fixed decomposition lines ($\delta|_{1,3,7}$) [12] and O-Sort with incorporated clustering change sensing. In model B (error-aware model), the spike processor operates adaptively with the embedded frames [2].

The clustering performance of both error-affected and error-aware processors, is shown in Fig. 21 for a sequence of randomly selected input data. It demonstrates the clustering performance superiority of the adaptive spike processor under variable input signal conditions. Averaging the results in Fig. 21, yields an 84.5% median clustering accuracy for Model B compared to 73.3% for Model A.

Table II shows the automatic choice of the decomposition scaling factors (δ) for the 16 different combinations in Fig. 21, and details the training convergence time (also quantified in spike numbers W), number of iterations (NOIs) for defining the optimal threshold (T_{opt}) in the validation phase (t_2) and cluster change sensing time (CCST) for retraining initiation when there is cluster change in the recording path.

C. Case Study

This section provides a detailed multi-aspect analysis of Comb.(c, d'). Fig. 22(a) shows the clustering performance (CA_{CC}) versus the cluster mean convergence weight in (8). Ttransient average performance does not significantly change when the update weight W (NOSPC in Fig. 14) is higher than 35. The iterative-update procedure initially introduces error in cluster mean convergence and eventually converges to its true value. Fig. 22(b) shows the cluster border rotation in the sorting threshold (*SThr*) tuning phases 1-3; the cluster border is rotated by θ_1 and θ_2 degrees. This rotation is due to decreasing the initial value of *SThr* with a fixed step ($\Delta = 0.1$) in the validation phase (t_2). To qualitatively show the effectiveness of C-Sort, Fig. 22(c) shows the 2-D projection test [28] of two merging clusters and the improvement over three iterations. Moving towards T_{opt} is the same as separating merged clusters in feature space.

D. Comparison

Table III compares this work with other integrated spike processors featuring on-chip clustering. The processor in this paper is the first adaptive sorting chip that provides *on-chip parametric tunability* via the inclusion of the embedded frames (Frame 1 and Frame 2). Since the spike processor can be implemented in different technologies, a figure-of-merit (FOM) is required to characterize relative efficiencies. The proposed FOM relates the spike processor power dissipation to its performance and is defined as:

$$\text{FOM} = \frac{P_{\text{channel}}}{(\text{CA}_{\text{CC}}\cdot 100)\cdot \text{DF}|_{\text{Base/Scaling}}} \quad (\mu\text{W}) \qquad (10)$$

TABLE II
ERROR-AWARE SPIKE PROCESSOR PARAMETERS

Random Data	δ	Convergence Time	NOIs	CCST ($t > t_3$)
Comb.(a, a')	$\delta\|_{6,5,4}$	0.11 s (W = 12)	1	-
Comb.(b, c')	$\delta\|_{7,6,4}$	0.67 s (W = 29)	2 (M)	aa'→ bc' (1.8 s)
Comb.(b, d')	$\delta\|_{7,6,3}$	0.67 s (W = 29)	3 (M)	-
Comb.(c, b')	$\delta\|_{6,4,5}$	0.69 s (W = 32)	2(S)	bd'→cb' (1.92 s)
Comb.(d, a')	$\delta\|_{6,5,4}$	0.84 s (W = 36)	1	cb'→ da' (1.67 s)
Comb.(c, a')	$\delta\|_{6,5,4}$	0.48 s (W = 27)	1	da'→ ca' (1.88 s)
Comb.(a, b')	$\delta\|_{6,5,4}$	0.18 s (W = 13)	1	ca'→ ab' (1.95 s)
Comb.(b, a')	$\delta\|_{6,5,3}$	0.14 s (W = 11)	1	ab'→ ba' (2.15 s)
Comb.(d, b')	$\delta\|_{6,5,4}$	0.74 s (W = 36)	1	ba'→ db' (1.58 s)
Comb.(d, d')	$\delta\|_{7,6,2}$	0.98 s (W = 40)	3 (M)	-
Comb.(a, c')	$\delta\|_{6,4,5}$	0.25 s (W = 14)	2 (M)	dd'→ ac' (1.63 s)
Comb.(b, b')	$\delta\|_{6,7,3}$	0.23 s (W = 20)	2 (S)	ac'→ bb' (2.22 s)
Comb.(a, d')	$\delta\|_{6,4,5}$	0.41 s (W = 21)	3 (M)	bb'→ ad' (1.78 s)
Comb.(d, c')	$\delta\|_{5,7,6}$	0.83 s (W = 39)	1	ad'→ dc' (1.38 s)
Comb.(c, d')	$\delta\|_{7,6,2}$	0.92 s (W = 35)	3 (M)	dc'→ cd' (1.43 s)
Comb.(c, c')	$\delta\|_{6,4,5}$	0.61 s (W = 32)	2 (M)	-

Under column NOIs: M = Merge, S = Split.

[2] If only Frame 1 or only Frame 2 are used in the adaptive spike processor, the latter provides almost 5% higher median clustering accuracy compared to the former.

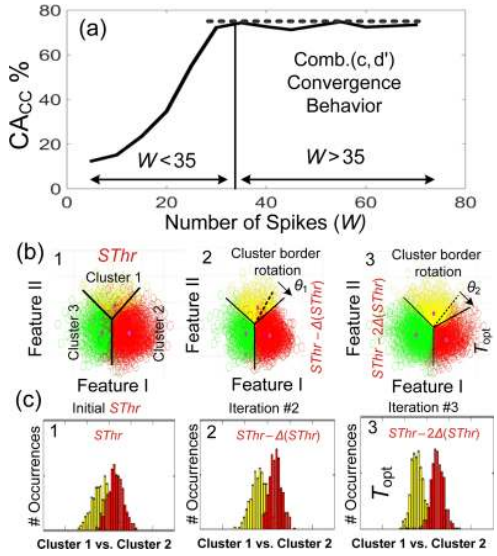

Fig. 22. (a) Cluster convergence illustration for Comb.(c, d') versus W (NOSPC in Fig. 14). $W = 35$ forms a reliable feature space for clustering and CA_{CC} saturates beyond this value. (b) Iteration phases. In each iteration the cluster border rotates by θ degrees. 2-D projection plots are based on clustered FVs in assignment phase for the duration of 30 seconds. (c) 2-D projection test for different iterations (1-3). The merging degrades with tuning the sorting threshold to T_{opt}.

where $P_{channel}$ is the power dissipation per channel, $CA_{CC}{\cdot}100$ is the clustering accuracy score, and $DF|_{Base/Scaling}$ is the downscaling factor which adjusts for dynamic power characteristics of the spike processor in different technologies where [29]:

$$DF|_{Base/Scaling} = \frac{\alpha \cdot N_t \cdot C_{avg} \cdot V_{sup}^2 \cdot f_{opt}|_{Base}}{\alpha \cdot N_t \cdot C_{avg} \cdot V_{sup}^2 \cdot f_{opt}|_{Scaling}} \qquad (11)$$

where α is the switching probability, N_t is the number of transistors in the design, C_{avg} is the MOSFET capacitance value, V_{sup} is the supply voltage in a particular technology, and f_{opt} is the operating clock frequency. For the spike processor *Base* is the reference technology and *Scaling* is the target technology. For example, for FOM evaluation of the spike processor in this work, from *Base* technology (180 nm, 1.8 V) to *Scaling* technology (65 nm, 0.27 V), both operating at the same f_{opt} and having the same αN_t factor, using (11) with $C_{avg}|_{Base} / C_{avg}|_{Scaling} = 2.7$, yields $DF|_{Base/Scaling} = 123$.

As reported in Table III, when the effect of different technology dimensions are accounted, this adaptive spike processor has 4.4X lower FOM compared with [13] and achieves almost 10% higher clustering accuracy in online clustering. Although the spike processor in [16] has approximately 23.3X lower FOM compared to the adaptive spike processor in this work, the latter achieves almost 13% higher clustering performance in unsupervised mode, which allows for accurate interpretation of neural activities.

V. Conclusion

An adaptive processing methodology has been introduced to enhance the performance of synchronous processing systems. It embeds reconfigurable sensing frames into the synchronous processing path that learn the characteristics of the variable input neural signals and adapts the functionality accordingly to improve the accuracy. As proof of concept, an adaptive spike processor has been designed, fabricated and evaluated. In addition, a configurable online sorting method (C-Sort) has been proposed which incorporates defining optimal threshold level (T_{opt}) and sensing active neurons in the recording channel. The chip prototype provides 84.5% accuracy and consumes 148 µW from a 1.8 V supply voltage. A dynamic testing methodology has been used to demonstrate the effect of signal model learning on clustering performance under variable conditions. Improved accuracy performance has been achieved compared to the state-of-the-art online clustering processors. The focus of future work will be towards the development of a multichannel spike sorting processor based on the adaptive processing methodology implemented in an advanced digital CMOS technology.

TABLE III
Comparison With Other Work

Reference	[13]	[16]	This Work (TW)
Detection	Absolute ($4\sigma_N$)	ICD [a]	*SThr* & ωNEO
Alignment	Slope	Peak	Peak
Feature Extraction	Spike Template	ICFE [b]	ADDs
Clustering	O-Sort	k-means	C-Sort
Compression Factor	240X	257X	150X/240X
CMOS Process (nm)	65	65	180
Supply Voltage (V)	0.27	0.54	1.8
Power (µW/channel)	4.68	0.175	148 [c]
Area (mm^2/channel)	0.07	0.003	2.7
Clustering Accuracy (CC_{AC})	[d] U: 97% M: 75% L: 45%	U: 83-99% M: 72-87% [e] L: 68-77%	U: 92.8% M: 84.5% L: 67%
$FOM|_{DF}$ (µW) $DF|_{Base/Scaling}$	624×10^{-4} $DF|_{[13]/[13]} = 1$	[f] 6.07×10^{-4} $DF|_{[16]/[13]} = 4$	142×10^{-4} $DF|_{[TW]/[13]} = 123$
Adaptive Design	No	No	Yes

(a) Integer coefficient detector (ICD).
(b) Integer coefficient feature extraction (ICFE).
(c) In 45-nm NAN-GATE the power consumption is 20 µW (1.1 V supply voltage).
(d) Upper (U), Mean (M), Lower (L).
(e) 87% average accuracy when the number of clusters are set manually.
(f) Based on the unsupervised clustering accuracy.

References

[1] A. Mohammed, M. Zamani, R. Bayford, and A. Demosthenous, "Towards on-demand deep brain stimulation using online Parkinson's disease prediction driven by dynamic detection," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 12, pp. 2441–2452, Dec. 2017.

[2] A. Beuter, J.P. Lefaucheur, and J. Modolo, "Closed-loop cortical neuromodulation in Parkinson's disease: An alternative to deep brain stimulation?," *Clin Neurophysiol*, vol. 125, no. 5, pp. 874–85, May 2014.

[3] T. S. Davis, H. Wark, and D. T. Hutchinson, "Restoring motor control and sensory feedback in people with upper extremity amputations using arrays of 96 microelectrodes implanted in the median and ulnar nerves," *J Neural Eng.*, vol. 13, no. 3, pp. 036001, Mar. 2016.

[4] T. Yanagisawa, M. Hirata, Y. Saitoh, T. Goto, H. Kishima, R. Fukuma, H. Yokoi, Y. Kamitani, and T. Yoshimine, "Real-time control of a prosthetic hand using human electrocorticography signals," *J. Neurosurg.*, vol. 114, no. 6, pp. 1715–1722, Jun. 2011.

[5] T. W. Bergeret al., "Restoring lost cognitive function: Hippocampal-cortical neural prostheses," *IEEE Eng. Med. Biol. Mag.*, vol. 24, no. 5, pp. 30–44, Sep./Oct. 2005.

[6] R. J. Vetter, J. C. Williams, J. F. Hetke, E. A. Nunamaker, and D. R. Kipke, "Chronic neural recording using silicon-substrate microelectrode arrays implanted in cerebral cortex," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 896–904, Jun. 2004.

[7] A. Rodriguez-Perez, J. Ruiz-Amaya, M. Delgado-Restituto, and A. Rodriguez-Vazquez, "A low-power programmable neural spike detection channel with embedded calibration and data compression," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 2, pp. 87–100, Apr. 2012.

[8] Y. Liu, S. Luan, I. Williams, A. Rapeaux, and T. G. Constandinou, "A 64-channel versatile neural recording SoC with activity-dependent data throughput," *IEEE Trans. Biomed. Circuits Syst.*, pp. 1–12, doi: 10.1109/TBCAS.2017.2759339, Nov. 2017.

[9] G. Buzsáki, A. Draguhn, "Neuronal oscillations in cortical networks," *Science*, vol. 304, no. 5679, pp. 1926–1929, Jun. 2004.

[10] H. G. Rey, M. J. Ison, C. Pedreira, A. Valentin, G. Alarcon, R. Selway, M. P. Richardson, and R. Quian Quiroga, "Single-cell recordings in the human medial temporal lobe," *J. Anat.*, vol. 227, no. 4, pp. 394–408, Oct. 2015.

[11] Y. Yang, S. Boling, and A. Mason, "A hardware-efficient scalable spike sorting neural signal processor module for implantable high-channel-count brain machine interfaces," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 4, pp. 743-754, Aug. 2017.

[12] V. Karkare, S. Gibson, and D. Markovic, "A 130 W, 64-channel neural spike-sorting DSP chip," IEEE J. Solid-State Circuits, vol. 46, no. 5, pp. 1214–1222, May 2011.

[13] V. Karkare, S. Gibson, and D. Markovic´, "A 75-µW, 16-channel neural spike-sorting processor with unsupervised clustering," *IEEE J. Solid-State Circuits*, vol. 48, no. 9, pp. 2230–2238, Sep 2013.

[14] T.-C. Chen, W. Liu, and L.G. Chenet, "128-channel spike sorting processor with a parallel-folding structure in 90 nm process," in *Proc. ISCAS*, Taipei, Taiwan, 2009, pp. 1253–1256.

[15] T.-T. Liu and J. M. Rabaey, "A 0.25 V 460 nW asynchronous neural signal processor with inherent leakage suppression," *IEEE J. Solid-State Circuits*, vol. 48, no. 4, pp. 897–906, Apr. 2013.

[16] S. M. A. Zeinolabedin, A. T. Do, D. Jeon, D. Sylvester, and T. T. Kim, "A 128-channel spike sorting processor featuring 0.175 µW and 0.0033 mm^2 per channel in 65-nm CMOS," *VLSI-Circuits Dig. Tech. Papers*, Honolulu, HI, Jun. 2016.

[17] Y. Yuan, C. Yang, and J. Si, "The m-sorter: an automatic and robust spike detection and classification system," *J. Neurosci. Methods,* vol. 210, no. 2, pp. 281–290, Sep. 2012.

[18] M. Lewicki, "A review of methods for spike sorting: The detection and classification of neural action potentials," *Network*, vol. 9, no. 4, pp. R53–78, Nov. 1998.

[19] M. Zamani, D. Jiang, and A. Demosthenous, "A highly accurate spike processor with reconfigurable embedded frames for unsupervised and adaptive analysis of neural signals," in *Proc. ESSCIRC*, Leuven, Belgium, Sep. 2017.

[20] M. Zamani and A. Demosthenous, "Feature extraction using extrema sampling of discrete derivatives for spike sorting in implantable upper limb neural prostheses," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 4, pp. 716–726, Jul. 2014.

[21] M. Zamani, and A. Demosthenous, "Dimensionality reduction using asynchronous sampling of first derivative features for real-time and computationally efficient neural spike sorting," in *Proc. ICECS,* Abu Dhabi, United Arab Emirates, Dec. 2013, pp. 237–240.

[22] Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural. Comp.*, vol. 16, no. 8, pp. 1661–1687, Aug. 2004.

[23] S. Mukhopadhyay and G. Ray, "A new interpretation of nonlinear energy operator and its efficacy in spike detection," *IEEE Trans. Biomed. Eng.*, vol. 45, no. 2, pp. 180–187, Feb. 1998.

[24] 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, no. 1-2, pp. 204–224, Jun. 2006.

[25] H. W. Lilliefors, "On the Kolmogorov-Smirnov test for normality with mean and variance unknown," *J. Amer. Statist. Assoc.*, vol. 62, no. 318, pp. 399–402, Jun. 1967.

[26] [Online]Available: https://www2.le.ac.uk/centres/csn/software/software

[27] D. Bol, R. Ambroise, D. Flandre, and J.-D. Legat, "Impact of technology scaling on digital subthreshold circuits," Symposium on VLSI (ISVLSI'08), Montpellier, France, 2008, pp. 179–184.

[28] C. Pouzat, O. Mazor, and G. Laurent, "Quality metrics to accompany spike sorting of extracellular signals," *J. Neurosci. Methods*, vol. 31, no. 24, pp. 8699–8705, Jun. 2011.

[29] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," Proc. IEEE, vol. 83, no. 4, pp. 498–523, Apr. 1995.