Frameworks for Efficient Brain-Computer Interfacing

Daniel Valencia, Jameson Thies, and Amirhossein Alimohammad
Department of Electrical and Computer Engineering
San Diego State University, San Diego, U.S.A

***Abstract*—One challenge present in brain-computer interface (BCI) circuits is finding a balance between real-time on-chip processing in-vivo and wireless transmission of neural signals for off-chip in-silico processing. This article presents three potential frameworks for investigating an area- and energy-efficient realization of BCI circuits. The first framework performs spike detection on the filtered neural signal on a brain-implantable chip and only transmits detected spikes wirelessly for offline classification and decoding. The second framework performs in-vivo compression of the on-chip detected spikes prior to wireless transmission for substantially reducing wireless transmission overhead. The third framework performs spike sorting in-vivo on the brain-implantable chip to classify detected spikes on-chip and hence, even further reducing wireless data transmission rate at the expense of more signal processing. To alleviate the on-chip computation of spike sorting and also utilizing a more area- and energy-effective design, this work employs, for the first time, to the best of our knowledge, an artificial neural network (ANN) instead of using relatively computationally-intensive conventional spike sorting algorithms. The ASIC implementation results of the designed frameworks are presented and their feasibility for efficient in-vivo processing of neural signals is discussed. Compared to the previously-published BCI systems, the presented frameworks reduce the area and power consumption of implantable circuits.**

I. Introduction

The brain continuously controls an extraordinary amount of bodily function. Voluntary movement, for example, is primarily controlled through the activity of nerve cells, known as neurons, in the motor cortex region of the brain. For voluntary movement, the brain's neural signals are sent down the spinal cord to control the contraction of certain muscles. However, spinal cord injury and other neurodegenerative diseases can impede the natural ability of the brain to communicate through the neural pathways. To restore bodily functions, brain-implantable integrated circuits have been used to record and potentially decode neural signals. Brain-computer interfaces (BCIs) refers to the microsystems capable of recording neural signals, performing signal processing, and transmitting data off-chip for further processing, thus offering a secondary pathway for neural signals. Each BCI channel typically samples at around 20 – 24 KHz [1], with 8 to 24 bits resolution for the analog-to-digital (ADC) converter [2], [3] to avoid losing high frequency elements of neural signals. A system with 64 channels, each channel sampling at 20 KHz, creates 1.28 MSamp./second. This corresponds to output data rates ranging from 10.24 to 192 Mbps and correlates to power consumption rates in the mW range [4]–[8], which imposes serious limitations, such as the potential threat of heat-related tissue damage. A trade-off is inevitably present between the power consumption of on-chip neural signal processing and that of wireless data transmission for off-chip classification and decoding. An important question thus arises regarding whether it is more efficient to process neural data on-chip and transmit only compressed or classified spikes, or is it more efficient to transmit data after minimal or partial processing? By efficiently performing neural signal processing directly at the recording site using braing-implantable integrated circuits, the wireless transmission rates can be drastically reduced, thus relaxing the requirements on the wireless transmission. To evaluate the efficiency of the BCI system configurations, three frameworks, as shown in Fig. 1, are designed, implemented, and compared. Framework I uses a spike detector in-vivo to reduce the data transmission by sending only detected spikes for off-chip classification and decoding. The second potential framework for minimizing energy is to apply spike compression on a brain-implantable chip. By exploiting the sparsity of neural signals, the data rate, and hence, energy dissipation, through the skull can be substantially reduced. Framework III performs spike sorting on-chip to even further reduce wireless transmission of data for off-chip decoding at the expense of more on-chip processing in-vivo. Spike sorting is the process of separating the activities of individual neurons, i.e., determining which neural action potentials, referred to as spikes, correspond to which neurons. Because the computational and energy requirements of certain spike sorting algorithms may not be suitable for area- and power-constrained applications, alternative approaches must be exploited and their relative area and energy dissipation must be analyzed. The main goal of the three BCI frameworks shown in Fig. 1 is thus to investigate an optimal balance between on-chip neural signal processing and wireless data transmission such that the total energy dissipation of the brain-implantable chip is minimal.

The rest of this article is organized as follows. Section II presents the spike detection, compression, and sorting schemes utilized in Frameworks I, II, and III, respectively. The hardware architecture of the spike detection, compression, and sorting are presented in Section III. Section IV presents the ASIC characteristics and implementation results of the three designed frameworks. The implemented frameworks are also compared with the state-of-the-art ASIC implementations of BCI systems and the feasibility of the presented frameworks for in-vivo signal processing is discussed. Finally, Section V makes some concluding remarks.

II. Framework Configurations

A. Spike Detection

Instead of transmitting sampled raw neural signals for off-chip (in-silico) signal classification and decoding, which

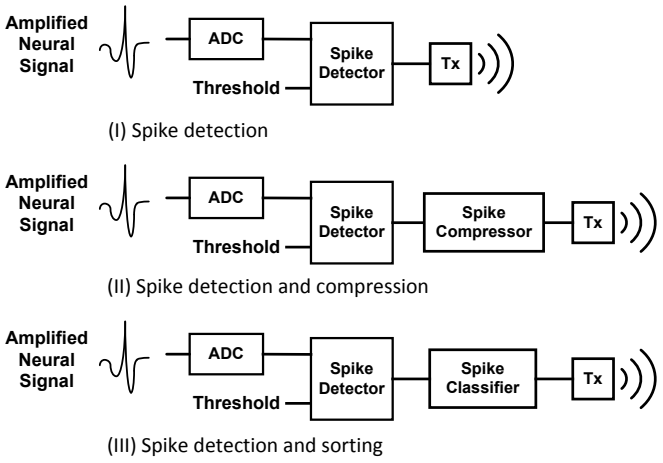

Fig. 1. Potential frameworks for area- and energy-efficient BCI systems.

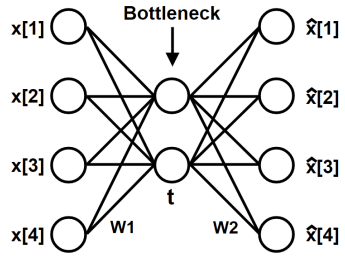

Fig. 2. An autoencoder with $N = 4$ and $M = 2$. The hidden layer between the input and output layers is considered the bottleneck, which is the point where the data is coded. Going from one layer to the next requires weight matrix multiplication, bias vector addition, and activation function computation.

could be unacceptable for BCI systems with a relatively large number of recording channels, one approach to reduce the output data rate, and hence, wireless transmission energy dissipation, is to detect neural action potentials (spikes) on a chip in-vivo and only transmit detected spikes wirelessly for further processing in-silico. An action potential is defined as the spiking activity of neurons, and this level of recording is known as single unit recording. This approach is shown in Framework I in Fig. 1. Spike detection is a two-step process in which action potentials are identified from ambient neural noise. First, a pre-emphasis operation is performed to prepare the signal for the second step, which is thresholding. Various algorithms have been proposed for pre-emphasizing neural signals, such as the absolute value operator [9], the non-linear energy operator (NEO) [10], and the discrete wavelet transform (DWT) [11]. In the thresholding step, the pre-emphasized signal is then compared to a threshold value. The presence of a spike can be inferred when the pre-emphasized signal crosses the threshold. For simpler hardware implementation, some designs avoid pre-emphasizing the neural signal and the thresholding is performed directly on the signal, which is referred to as the voltage threshold detection method [12]. By avoiding pre-emphasis, the voltage threshold detection method can be implemented without using multipliers, which lends itself to an area-efficient hardware implementation. However, for a more robust spike detection, we have designed and implemented the NEO algorithm. NEO is defined as:

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

where $x[n]$ denotes the neural signal at time n. The result is only large when the signal is both large in power ($x[n]^2$) and in frequency ($x[n]$ is large while both $x[n-1]$ and $x[n+1]$ are small).

B. Spike Compression

While Framework I only transmits detected spike waveforms, a significantly larger reduction in output data rate can be obtained by both performing spike detection as well as reducing the number of samples required for representing the spike waveform, i.e., compressing the spike waveforms on-chip in-vivo as shown in Framework II in Fig. 1. One important design trade-off in compression techniques for BCIs is the power consumption required for desirable signal reconstruction quality. Compressed sensing (CS) has shown as being a viable method for the compression of neural spikes [13]–[16]. While CS algorithms offer high-quality reconstruction of compressed signals, they require a relatively large number of operations. An alternative approach is to utilize an autoencoder [17], which is a type of artificial neural network (ANN) trained so that the network's output reproduces its input. Fig. 2 shows the signal flow graph (SFG) of a basic undercomplete autoencoder. The first layer transforms the four inputs down to two inputs. This part of the network is referred to as the bottleneck, and is the point at which the data is coded (compressed). The output layer of the SFG shows how the compressed bottleneck outputs are used to reconstruct the original input vector. $w1$ and $w2$ denote the transformation matrices used to downsize the input vector to the bottleneck output size, and to reconstruct the input vector from the output of the bottleneck, respectively.

By placing constraints on the parameters of the autoencoder, significant features of the dataset can be learned during the network's training. The fundamental building block of an ANN is a neuron. Neurons typically accept a data vector $\mathbf{x}$ from either the previous layer or the inputs, perform element-wise multiplication with a weight matrix $\mathbf{W}$, sum the products with a bias $\mathbf{b}$, and then pass the result through an activation function $f(\sum w_{i,j}x_i + b_i)$. In addition to input and output layers, neurons constitute hidden layers. The input layer simply accepts the input vector $\mathbf{x}$ and passes it to the first hidden layer. Hidden layer computation can be defined as a matrix multiplication, vector addition, and applying an activation function as:

$$y = f\left(\begin{bmatrix} w_{1,1} & \dots & w_{1,N} \\ \vdots & \ddots & \vdots \\ w_{M,1} & \dots & w_{M,N} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix}\right),$$

where M and N denote the number of neurons in the hidden and input layer, respectively. Generally the output layer processes data in the same way as a hidden layer. One commonly used activation function at the output layer is the Softmax function [18], defined as $f(x_i) = e^{x_i} / \sum e^{\mathbf{x}}$, which provides the probabilities of each output classification. In general, the

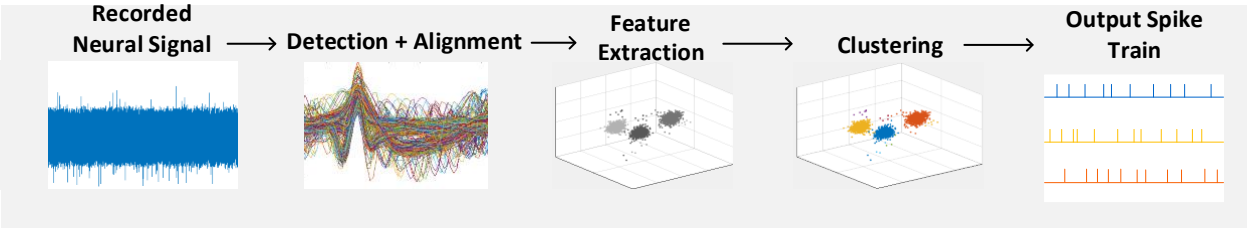

Fig. 4. The conventional processing steps for offline spike sorting.

input and output of the autoencoder must be the same size N. The hidden layer of size M in an autoencoder, known as the bottleneck, is responsible for compression. Sending data through an M-sized layer gives the compression ratio (CR) N/M. The two primary elements of an autoencoder are the encoder and decoder. The bottleneck and the layers preceding it are referred to as the encoder and the remaining layers form the decoder, which completes the neural network computation to reconstruct an approximation of the input vector.

Fig. 3 shows the system-level block diagram of an autoencorder. For compressing a spike from N data points to M data points, an $M \times N$ weight matrix $\mathbf{W}_1$ is multiplied with the $N \times 1$ vector representing the spike waveform. The weight matrix is obtained via training and its values are stored on-chip. The autoencoder can be trained to compress and reconstruct a given type of data based upon what optimal compression and expansion are learned from the data. For compressing neural spikes, the designed autoencoder has a 64-node input layer, and one hidden layer with 4 nodes. This results in a 16 times compression ratio. Because the autoencoder has one hidden layer, the data is entirely compressed by the first matrix multiplication. Thus, the only computation for on-chip compression is a matrix multiplication. The matrix-vector multiplication requires $M \times N$ multiplications and $M \times (N-1)$ additions. The resulting $M \times 1$ compressed neural spike is then transmitted off-chip. As shown in Fig. 3, the off-chip reconstruction requires two vector additions, two activation function computations, and a matrix multiplication.

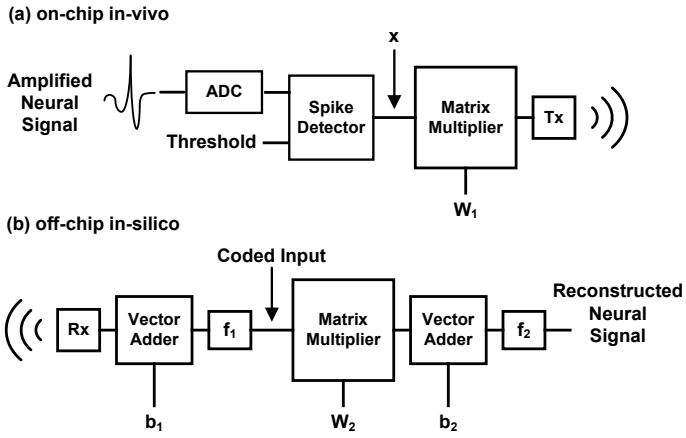

Fig. 3. System-level block diagram of an autoencoder for spike compression.

C. Spike Sorting

To even further reduce the transmitted data, an alternative approach is to perform spike sorting in-vivo. Spike sorting is the process of identifying spike waveforms and attributing them as being spikes emitted by specific neurons [12]. The reduction in data rate associated with spike sorting comes from the fact that the transmitted output is a small binary identifier used to encode which group, or class, of neuron the spike was emitted by. Fig. 4 shows the traditional processing steps for off-line spike sorting. First, spikes are detected from ambient noise in recorded neural signals. Spike detection is often performed with the aforementioned voltage threshold method, the absolute value method, or the NEO-based detection method. The spikes are then aligned to a particular metric, such as the point within the spike that has the highest amplitude or highest slope, as it has been shown that the alignment of the spike waveform can be crucial for the classification of spikes [19]–[21]. Depending on the detection method used, other alignment metrics such as maximum energy can also be used. Once spikes have been detected and aligned, feature extraction is employed. Feature extraction is a process in which certain features of the detected spike waveforms are selected and used as descriptors for those types of spike waveforms [12]. The number of features used is fewer than the number of samples used for the representation of the spike waveform, thus performing dimensionality reduction. Principal components analysis (PCA) [22] is a commonly employed algorithm for finding discriminatory features present in a collection of detected spike waveforms. However, because of PCA's computational complexity, it is often not considered for implementation on a brain-implantable chip, and more computationally-efficient feature extraction methods, such as discrete derivatives [23] and the integral transform [24], are considered for in-vivo realizations. The features can then be visualized in a k-dimensional space, where k denotes the number of selected features. In this k-dimensional space, clusters are formed, and the average of the cluster coordinates can be used to represent that cluster. Various clustering methods have been utilized for classifying spike features. Manual cluster cutting [12] involves plotting the features in a scatter plot followed by the manual formation of clusters by an operator. In k-means clustering [25], an operator first defines k clusters with random cluster centroids. The detected spikes are then assigned to the closest cluster centroid by way of a distance metric. The cluster centroids are then recalculated with every assignment. Finally, each detected spike is assigned to a cluster and an output spike train is reconstructed, which is a visual

representation of both the times of neural spiking as well as a cluster's rate of spiking.

Other spike sorting approaches, such as the template matching [26] and OSort [27] algorithms, eliminate the requirement of the computationally-intensive feature extraction step altogether. These methods can be thought of as utilizing κ number of spike features, where κ denotes the number of samples used for representing a spike waveform. In template matching-based spike sorting, the average spike waveform of the clusters (shown in the clustering step in Fig. 4) can be used as template waveforms, which represent different spike shapes. Template matching then uses a distance metric to quantify the similarity between the detected spikes and the stored templates. If the distance is within an acceptable margin, the spike is classified. While various methods, such as manual cluster cutting and k-means clustering, may be acceptable for off-line sorting, they require operator intervention. The OSort algorithm [27], however, continuously stores detected spikes into various clusters using cluster averages computed on-the-fly in an unsupervised fashion. As the clustering process continues, the cluster averages converge toward the optimal representation of the commonly occurring spike shapes as distinctly separable clusters. For efficient spike sorting realization, in this work we have taken an alternative approach to conventional methods and designed an ANN specifically for spike classification. This approach is similar in that each sample of the waveform can be considered as a feature that is passed to the classification ANN. The length of the input vector to the ANN and the number of spike classes define the size of the input and output layer, respectively. For our chosen application, 64 samples were sufficient to hold an entire spike, meaning that the input layer has 64 nodes. For three spike classes, the output layer of the designed sorting ANN has three nodes. The ANN offers a "best guess" as to which of the three spike classes the spike belongs to. To reduce overall system complexity, the designed ANN has one hidden layer with only one node.

Once the ANN topology has been established, it must be trained for a given task to find optimal weight and bias values. This is typically achieved through the process of backpropagation [28], which adjust weights and biases to minimize a loss function. The designed ANNs for compression and sorting were trained and tested using six datasets with different levels of noise and classification difficulty from the publicly-avialable Wave_Clus dataset [29]. To train the ANNs, the spikes from each dataset were aligned to maximum amplitude and divided into training and testing data. For ANNs trained on data with *Easy* and *Difficult* classification difficulties, there were 3,012 and 2,984 spikes, respectively. A network was trained for every combination of parameters, including classification difficulty, noise, and amount of data used for training. Ten networks for each combinations of parameters were trained using stochastic gradient descent [28] over 400 epochs, then tested using 14-bit signed fixed-point arithmetic with 6 integer bits and 8 fraction bits. Table I gives the accuracy of the sorting ANNs for various datasets and training characteristics. The best of the ten ANN classifiers for each combination of parameters is given in each row of Table I. It can be seen that increasing the amount of training data has a small effect on the sorting accuracy, but the ANNs still perform well with as little as 30% of data used for training.

TABLE I
ACCURACY OF THE SPIKE SORTING ANNS

Classification difficulty	*Noise standard deviation*	*Percent of data used for training*	*Classification accuracy*
Easy	0.05	30 %	99.91 %
		70 %	99.78 %
	0.1	30 %	99.48 %
		70 %	99.67 %
	0.2	30 %	97.44 %
		70 %	99.00 %
Difficult	0.05	30 %	98.66 %
		70 %	99.20 %
	0.1	30 %	99.19 %
		70 %	99.66 %
	0.2	30 %	97.56 %
		70 %	98.44 %

III. HARDWARE ARCHITECTURES OF THE THREE FRAMEWORKS

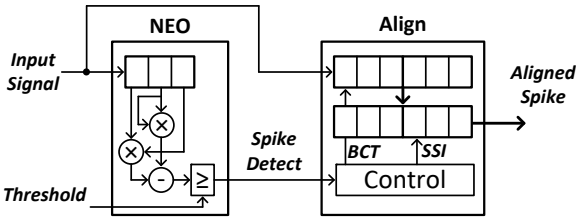

Fig. 5. Block diagram of the designed NEO-based spike detection and alignment unit.

A. Spike detector

The spike detection unit is utilized in all three BCI Frameworks. For assessing the accuracy and feasibility of hardware architectures, we have implemented both the voltage threshold and NEO-based spike detection schemes. The implementation of the voltage threshold scheme is simple, as it only uses a comparator to determine if the value of the input signal has crossed a variable threshold value. The block diagram of the designed NEO-based spike detection and alignment unit is shown in Fig. 5. The detection unit implements the NEO Equation (1) and the alignment unit implements the maximum amplitude spike alignment using fixed-point arithmetic. The filtered neural signal is passed to a three sample shift register, which stores samples $x[n-1]$, $x[n]$, and $x[n+1]$ required in Equation (1). The NEO output $\psi(x[n])$ is then computed and compared to a given *Threshold*. If the energy of the signal is greater than or equal to the *Threshold*, the comparator's output is asserted high, which is used as a spike detect signal monitored by the *Control unit* in the alignment module. The alignment module consists of two buffers, the upper one which is a serial-in, parallel-out (SIPO) shift register to buffer the neural waveform, and the lower one which is a parallel-in, parallel-out (PIPO) register of the same size. The *Control unit* enables the transfer of data from the SIPO register to the

PIPO register via the *BCT* buffer content transfer signal when the *Spike Detect* signal is asserted. The *Control unit* then spans through the PIPO register in search of the maximum index via the search span index signal *SSI*. Once the maximum index is found, the *Control unit* performs a part select to output a 64-sample waveform from the PIPO register via the output port *Aligned Spike*.

B. Spike Compression

The block diagram for the autoencoder-based compression scheme is shown in Fig. 6. An array of M multiply-accumulate (MAC) units accept the inputs in parallel and multiply each of the 64-inputs by a weight corresponding to the inputs index within the spike waveform. To feed the weights to the MACs, a series of M read-only memories (ROMs) are used.

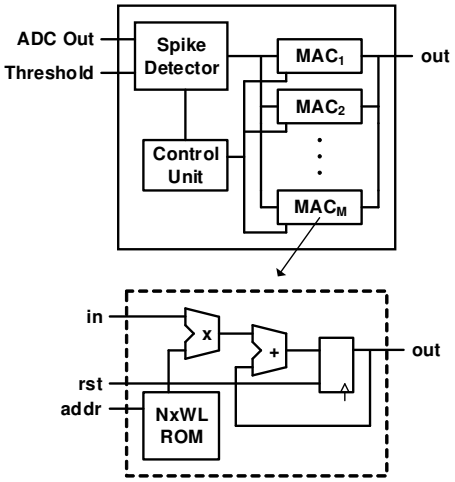

Fig. 6. The block diagram of the designed autoencoder-based compression unit.

The spike detector serially feeds the detected spike to η different MAC units, where η denotes the desired compression ratio. One counter is used as an index to four ROMs, each of which holds one row of the weight matrix. After 64 clock cycles, the latches in the MAC units each hold one of the four individual values in the output vector. We implemented this circuit using 14-bit fixed-point arithmetic. Because the spike waveform is compressed from 64 samples to 4 samples (i.e., 16 $\times$ compression), and each sample contains 14 bits, the output of the compression unit is 56 bits. A flag is added indicating the presence of a spike, which increases the output of the compression circuit to 57 bits per spike.

C. Spike Sorting

The spike sorting ANN requires the computation of input, hidden, and output layers, as well as additional output logic to determine spike's class based on the values at the output layer. A generic equation for the computation of each of the three output values can be given as:

$$y_n = f\left(\begin{bmatrix} w1_{1,1} & \dots & w1_{1,64} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{64} \end{bmatrix} + b1_1 \right) w2_{1,n} + b2_n,$$

where **x** denotes the input vector, **W1** and **b1** denote the weights and biases of the hidden layer, respectively, $f(\cdot)$ denotes the activation function, **W2** and **b2** denote the weights and biases of the output layer, respectively, and y_n denotes the result at node n of the output layer. We chose to use a Rectified Linear Unit (ReLU) activation function, $f(x) = \max(x, 0)$, to reduce circuit complexity while maintaining high classification accuracy [30].

Fig. 7 shows the processing element (PE) of the ANN-based sorting circuit to calculate the three values y_1, y_2, and y_3 at the output layer, which are used to determine the class of spikes. Over the first 64 clock cycles of processing a spike, the PE of the spike sorting ANN accepts the 64-length input vector **x** and multiplies it by the 1×64 weight matrix **W** using 64 multiplications and 63 additions. Note that the weight matrix is 1×64 since our designed ANN has only one node in the hidden layer. Since the weight matrix is constant and the spike vector **x** is passed one sample at a time to the ANN, the vector multiplication can be computed using a single MAC iteratively. On the 65-th clock cycle, the hidden node bias $b1$ is read from the ROM and added to $\mathbf{w1}_n \times$ **x**. The result is then passed through the ReLU unit and stored in a latch. The value in the latch is then multiplied by an output layer weight $\mathbf{w2}_n$ and added to an output layer bias $b2_n$ the following clock cycle. This is repeated two more times to compute the data at all output layer nodes. All weights and biases for both the hidden and output layers are stored in the ROM. Rather than using a complicated activation function at the output layer, such as Softmax [18], the output logic in this circuit simply gives the index of the latch containing the larger value. Not only does this reduce the complexity of the circuit, it also reduces the size of the output to only two bits. With an additional bit indicating a spike, this circuit has a 3-bit output. After 71 clock cycles, the spike class and a flag signaling the occurrence of a spike are produced at the output.

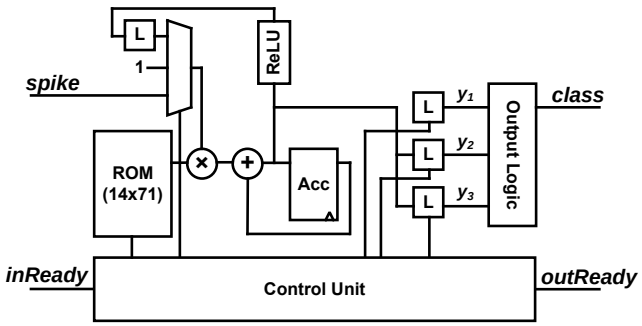

Fig. 7. Processing element of the spike sorting ANN.

For a comparative analysis, we have also implemented template matching [31] and OSort-based spike sorting schemes. Template-matching is a method for the classification of spikes based on pre-computed, stored templates. Because the template waveforms are generated off-chip on pre-recorded neural signals, complex algorithms can be employed to generate the ideal templates. The block diagram of the designed and implemented template matching-based spike sorting unit is shown in Fig. 8. The pre-computed templates are loaded into

the design during a programming phase and are stored in shift registers. The neural signals are represented using a 16-bit fixed-point representation, with 5 bits allocated for the integer portion and 11 bits allocated for the fractional part. An array of *EDU* units compute the Euclidean distance between the *Aligned Spike* and one of the stored templates. The *COMP* unit then finds the minimum of the Euclidean distances and compares it to the *Distance Threshold*, which denotes the maximum distance between two spikes to be considered as the same class of spike. If the minimum distance is less than the *Distance Threshold*, the *COMP* unit generates a 3-bit *Spike ID* to determine the class of detected spike. The number of bits used to represent the spike class is determined by the number of templates.

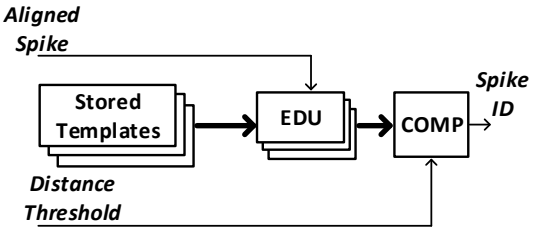

Fig. 8. Block diagram of the template matching-based spike sorting unit.

OSort can be considered as a learning-scheme for template matching-based spike sorting. In fact, OSort was developed to aid in the study of neural signals for in-lab experiments in which each possible spike waveform is saved for analysis [27]. Doing so, OSort can constantly adapt cluster average waveforms and can provide an ongoing estimate of the detected waveforms found in the neural signal. Compared to the static nature of template-matching, in which the templates may be fixed, OSort can thus adapt to both noise variations in the environment or electron-drift, which may both distort the waveform shape. OSort considers every sample of the spike waveform as a feature in a λ-sample feature space, where λ denotes the number of samples to represent a spike waveform. This is an example of a pre-processing method that is commonly referred to as a supervised learning scheme, in which the processing of future data relies on the availability of prior data.

The block diagram of the designed OSort-based spike sorting unit is shown in Fig. 9. The inputs to the clustering unit are the *Aligned Spike* and the assignment and clustering thresholds Th_A and Th_C, respectively. The samples of the spike waveforms are represented in 16-bit fixed-point format with 5 bits for the integer portion and 11 bits for the fractional part. The detected and aligned spike is compared to the cluster averages *Clus. Avg.* using the *Distance Unit* module, which computes the Euclidean distance. Similar to the template-matching unit, the *COMP* unit finds the minimum distance computed by the *Distance Unit* and compares it to the assignment threshold $Th_{\rm A}$. If the minimum distance between the aligned spike and one of the cluster averages is less than or equal to the assignment threshold $Th_{\rm A}$, the *Control Unit* generates a *Spike ID* and enables the memory unit *Cluster Mem.* to write the aligned spike into the matching cluster. The memory unit *Cluster Mem.* not only stores clusters and their assigned spikes, it also has the ability to perform cluster averaging and merging. The memory unit averages a cluster when 16 spikes have been assigned to it, which then updates the average of that particular cluster. When a cluster average is updated, the memory unit also compares the distance between the newly updated cluster and the other cluster averages. The minimum distance between the newly updated cluster average and a different cluster average is compared to the clustering threshold Th_C. If the distance does not exceed Th_C, the two clusters are merged.

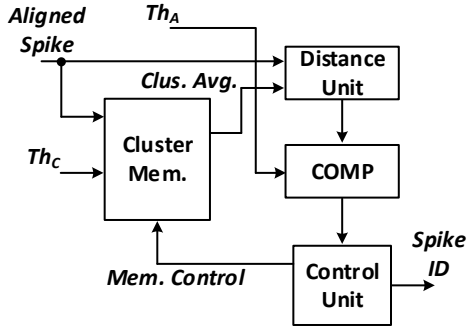

Fig. 9. Block diagram of the designed OSort module.

IV. Simulation and ASIC Implementation Results of the BCI Frameworks

For the evaluation of the designed BCI Frameworks, we use the publicly available Wave_Clus Database [29]. Within the database, datasets differ based on the ease of spike classification (*Easy* and *Difficult*) and noise. The database includes 20 different simulated neural waveforms, each of which contain three different classes of neural spikes. The simulated waveforms were created by randomly selecting spikes from a database of 594 spike shapes from neural recordings, then placed with random amplitudes and times in each waveform to resemble spikes from other neurons. Then, a preselected spike train with predefined types of spikes was imposed on the random signal. Noise with a standard deviation between 0.05 and 0.4 relative to the magnitude of the spikes is added to the simulated waveform. The data is first simulated at 96 KHz, then down sampled to 24 KHz. Spikes detected in the Wave_Clus dataset were divided into 30% used for training and 70% used for testing. We trained our designed autoencoder on the Wave_Clus dataset with *Easy* classification and 0.05 noise which had a -30.5019 signal-to-noise-ratio. The signal to noise ratio is defined as $10\log_{10}(V_{\rm RMS}/\sigma_n)$, where $V_{\rm RMS}$ denotes the RMS voltage of the signal and σ_n denotes the noise variance. The autoencoder was trained over 1,317 epochs. The network was then tested on the remaining 70% of the spikes. The reconstructed spike waveforms have an average signal-to-noise-and-distortion ratio (SNDR) of 19 dB, a standard deviation of 2.71 dB, and a worst case reconstruction of 7.13 dB. The SNDR is defined as $20\log_{10}(\mathrm{norm}(x)/\mathrm{norm}(x-\hat{x}))$, where $\mathrm{norm}(.)$ denotes the L2 norm, x denotes the original data, and $\hat{x}$ denotes the reconstructed data [32].

We implemented our designed circuits for detection, compression, and sorting using Synopsis DC Compiler in a stan-

TABLE III
THE ASIC CHARACTERISTICS AND IMPLEMENTATION RESULTS OF DIFFERENT BRAIN-COMPUTER INTERFACING SYSTEMS.

Work	Algorithm	Tech. (nm)	Core voltage (V)	Operating frequency (kHz)	Area per channel (μm^2)	Power per channel (μW)	Sorting Accuracy (%)	Data rate reduction	Transmission power per channel (μW)	Total power per channel (μW)
[33]	Feature Extraction	90	0.55	4000	60	2	77	11×	49	51
[34]	OSort	65	0.27	480	70	4.68	75	240×	2	6.68
[35]	Feature Extraction	45	1.1	960	2700	20	84.5	240×	3	23
[36]	Feature Extraction	130	1.2	160	23	0.75	–	–	–	–
[37]	BOTM	40	1.1	500	17.5	19	93	–	–	–
[38]	Feature Extraction	65	0.54	3200	3	0.175	86	257×	2	2.175
Ours [31]	Template Matching	45	0.25	24	300	0.064	90	3200×	0.36	0.42
Ours	OSort	32	1.16	24	2570	2.78	87	1600×	0.72	3.5
Ours	Spike Detection	32	0.7	20	3	0.58	–	6.24×	145.5	146.08
Ours [39]	Autoencoder Compression				13	1.84	–	98.24×	8.55	10.39
Ours	Spike Sorting ANN				9.2	1.04	98	1866×	0.45	1.49

dard 32-nm CMOS process with a global operating voltage of 0.7 V. Because neural recording devices often sample at 20 KHz, 20 KHz and 1.28 MHz are sufficient frequencies to process data from one and 64 channels, respectively. Table II gives the characteristics and ASIC implementation results of the designed circuits. Power consumption was reported by Synopsis DC Compiler using a Non-Linear Delay Model [40]. Cadence Encounter was used to place and route the synthesized netlist of the designed circuits. Fig. 10 shows the chip layouts of the compression circuit and sorting circuit including the detector. The autoencoder compression and the spike sorting ANN circuits occupy 13 μm^2, and 9.2 μm^2 of silicon area, respectively. Note that there is a 5μm buffer surrounding the chip layouts, which are not included in the area specifications given in Table II.

TABLE II
ASIC IMPLEMENTATION RESULTS OF THE THREE DESIGNED BCI FRAMEWORKS

Framework	*Frequency*	*Power (μW)*	*Area (μm^2)*
Spike Detection	20 (kHz)	0.58	3
Autoencoder Compression		1.84	13
Spike Sorting ANN		1.04	9.2
Spike Detection	1.28 (MHz)	1.93	3
Autoencoder Compression		3.87	13
Spike Sorting ANN		2.82	9.2

Table III gives the ASIC characteristics and implementation results of various state-of-the-art BCI systems. For a fair comparison, we have listed the ASIC results for single-channel operation. The work in [33] implements NEO-based spike detection, aligns detected spikes to maximum derivative, and implements feature extraction utilizing the discrete derivatives algorithm. The design supports up to 64 channels via four 16-channel modules. The design in [34] uses the absolute value spike detection scheme and implements OSort clustering for 16 channels. The circuit in [35] performs single-channel sorting with NEO-based spike detection and discrete derivative feature extraction. It supports an unsupervised learning phase, similar to OSort-based implementations. The design presented in [36] is a multi-channel spike sorting ASIC based on feature extraction. The design utilizes decision trees for sorting in place of memory units. The work in [37] implements Bayes optimal template matching (BOTM) with support for unsupervised learning. The multi-channel spike sorting ASIC in [38] implements integer coefficient feature extraction and clustering. For a comparative analysis of data rate reductions, area utilizations, and power consumptions, we have implemented both the template-matching sorting in [31] and OSort-based clustering ASICs. Both of the designed ASICs utilize NEO-based spike detection and maximum amplitude alignment.

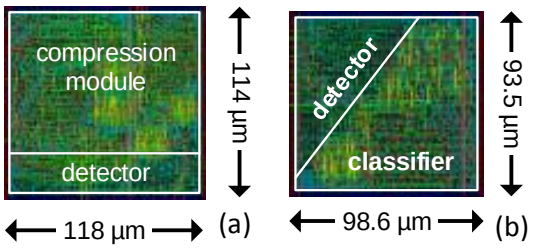

Fig. 10. Chip layouts of the (a) compression circuit for $M = 4$ and (b) ANN sorting circuit.

The latency of on-chip processing is also an important consideration. It is shown that different synaptic modifications can be produced by applying stimulation within 20 ms before or after pre-synaptic activation [41]. For classification/sorting of neural spikes, the processing latency can be defined as the number of clock cycles required after a spike is detected. The autoencoder compression takes 64 clock cycles to compress the spike waveform, not counting the off-chip processing latency. The spike sorting ANN, the template matching-based sorting scheme,and the OSort-based clustering implementations have processing latencies of 71, 64, and 32 clock cycles, respectively. While these systems would be operated at speeds

higher than the sampling rates, their latencies are well within the acceptable delay even if running at the sampling rate (20 – 24 kHz). The autoencoder compression latency correlates to 2.7 ms, while the spike sorting ANN, template matching-based implementation, and the OSort-based clustering units have processing latencies of 3 ms, 2.7 ms, and 1.3 ms, respectively. Unfortunately, the processing latency of the designs in [33]–[37] was not reported. The authors of [38] have reported the clock latency of the feature extraction and dimensionality reduction process, but they have not accounted for the clustering latency.

The accuracy of the spike sorting/classification implementations, defined as the percentage of correctly sorted spikes against those existing in the ground truth dataset [29], are reported in Table III. Note that our template matching-based ASIC [31], as well as the work in [35], [37], [38], utilize the same synthetic ground-truth dataset given in [29]. The work in [33] uses both real neural data and synthetic data to quantify the accuracy of their system, however, the synthetic data set is not the same as that used in [29]. The work in [34] and [36] both use only real data. The accuracy of the spike detection and autoencoder-based compression frameworks are not listed in Table III as these schemes do not perform on-chip spike waveform classification. Compared to the other state-of-the-art designs listed in Table III, it can be seen that our spike sorting ANN is among the most area- and power-efficient designs. From the results given in Table III we can conclude that for in-vivo area- and power-constrained processing of neural signals, it is preferred to implement the on-chip spike sorting ANN over other realizations. This is in part because the power consumption of the on-chip neural signal processing itself is relatively small compared to that of wireless transmission [42]. The transmission power consumption rates are estimated by multiplying the output data rates by a transmission power rate of 3 nJ per bit [43]. For a fair comparison, all of the metrics listed in Table III are for single-channel transmission. It follows then that a higher reduction in output data rate leads to a reduction of overall power consumption. By reducing the required output data rate, the total power dissipation of the BCI system can be decreased significantly, meeting tissue-safe thermal power constraints of brain-implantable devices [44].

V. Conclusion

This article presented the design and implementation of three brain-computer interface (BCI) frameworks. While each presented framework can reduce the data rate compared to raw neural signal sampling, the conclusion is made that the ideal framework for decreasing overall power consumption, including wireless transmission power, is the one that efficiently implements spike sorting using an artificial neural network on a brain-implantable chip. The ASIC implementation results of each framework were compared to state-of-the-art BCI systems. We have discussed the feasibility of our designed ASICs and confirmed that the power consumption rates are within the tissue-safe requirements for in-vivo processing of neural signals. It was shown that the overall power consumption is dominated by the output data rate of the system, and therefore the reduction in data rate is among the most important metrics in terms of overall power consumption reduction for in-vivo processing.

Acknowledgment

This work was supported by the Center for Neurotechnology (CNT), a National Science Foundation Engineering Research Center (EEC-1028725).

References

[1] M. Ballini *et al.*, "A 1024-channel CMOS microelectrode-array system with 26,400 electrodes for recording and stimulation of electro-active cells in-vitro," in *Symp. on VLSI Circuits*, 2013, pp. C54–C55.

[2] R. Shulyzki *et al.*, "320-channel active probe for high-resolution neuromonitoring and responsive neurostimulation," *IEEE Trans. on Biomedical Circuits and Systems*, vol. 9, no. 1, pp. 34–49, 2015.

[3] S. Yoshimoto *et al.*, "Implantable wireless 64-channel system with flexible ECoG electrode and optogenetics probe," in *IEEE Biomedical Circuits and Systems Conference*, 2016, pp. 476–479.

[4] A. Hennessy and A. Alimohammad, "Design and implementation of a digital secure code-shifted reference uwb transmitter and receiver," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 7, pp. 1927–1936, 2017.

[5] H. Bahrami, S. A. Mirbozorgi, L. A. Rusch, and B. Gosselin, "Integrated UWB transmitter and antenna design for interfacing high-density brain microprobes," in *IEEE International Conference on Ubiquitous Wireless Broadband*, 2015, pp. 1–5.

[6] H. Miranda and T. H. Meng, "A programmable pulse UWB transmitter with 34% energy efficiency for multichannel neuro-recording systems," in *IEEE Custom Integrated Circuits Conference*, 2010, pp. 1–4.

[7] A. Ebrazeh and P. Mohseni, "30 pj/b, 67 Mbps, centimeter-to-meter range data telemetry with an IR-UWB wireless link," *IEEE transactions on biomedical circuits and systems*, vol. 9, no. 3, pp. 362–369, 2014.

[8] J. L. Bohorquez, J. L. Dawson, and A. P. Chandrakasan, "A 350μw CMOS MSK transmitter and 400μw OOK super-regenerative receiver for medical implant communications," in *IEEE Symposium on VLSI Circuits*, 2008, pp. 32–33.

[9] I. Obeid and P. D. Wolf, "Evaluation of spike-detection algorithms for a brain-machine interface application," *IEEE Trans. on Biomedical Engineering*, vol. 51, no. 6, pp. 905–911, 2004.

[10] J. F. Kaiser, "On a simple algorithm to calculate the 'energy' of a signal," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1990, pp. 381 – 384.

[11] K. H. Kim, S. J. Kim, "A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio," *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 8, pp. 999 – 1011, 2003.

[12] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Computation in Neural Systems*, vol. 9, no. 4, pp. R53–R78, 1998.

[13] X. Liu et al., "A fully integrated wireless compressed sensing neural signal acquisition system for chronic recording and brain machine interface," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 4, pp. 874–883, 2016.

[14] W. Zhao, B. Sun, T. Wu, and Z. Yang, "On-chip neural data compression based on compressed sensing with sparse sensing matrices," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 242–254, 2018.

[15] Y. Suo, J. Zhang, T. Xiong, P. S. Chin, R. Etienne-Cummings, and T. D. Tran, "Energy-efficient multi-mode compressed sensing system for implantable neural recordings," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 5, pp. 0–0, 2014.

[16] J. Zhang et al., "Communication channel analysis and real time compressed sensing for high density neural recording devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 5, pp. 599–608, 2016.

[17] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[18] A. Sangari and W. Sethares, "Convergence analysis of two loss functions in soft-max regression," *IEEE Trans. on Signal Processing*, vol. 64, no. 5, pp. 1280–1288, 2016.

[19] R. S. LeFever and C. J. De Luca, "A procedure for decomposing the myoelectric signal into its constituent action potentials-part I: technique, theory, and implementation," *IEEE transactions on biomedical engineering*, no. 3, pp. 149–157, 1982.

[20] R. S. LeFever, A. P. Xenakis, and C. J. De Luca, "A procedure for decomposing the myoelectric signal into its constituent action potentials-part II: execution and test for accuracy," *IEEE transactions on biomedical engineering*, no. 3, pp. 158–164, 1982.

[21] M. S. Lewicki, "Bayesian modeling and classification of neural signals," in *Advances in Neural Information Processing Systems*, 1994, pp. 590–597.

[22] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[23] Z. Nadasdy, R. Q. Quiroga, Y. Ben-Shaul, B. Pesaran, D. A. Wagenaar, R. A. Andersen, "Comparison of unsupervised algorithms for on-line and off-line spike sorting," in *Proceedings of the Annual Meeting Soc. for Neuroscience*, 2002.

[24] A. Zviagintsev, Y. Perelman, and R. Ginosar, "Low-power architectures for spike sorting," in *International IEEE EMBS Conference on Neural Engineering.*, 2005, pp. 162–165.

[25] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, 1967, pp. 281–297.

[26] J. J. Capowski, "The spike program: A computer system for analysis of neurophysiological action potentials," in *Computer Technology in Neuroscience*, P. B. Brown, Ed. Washington: Hemisphere Publishing Corporation, 1976, Ch. 17, pp. 237–251.

[27] U. Rutishauser, E. Schuman, A. Mamelak, "Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo," *Journal of Neuroscience Methods*, vol. 154, pp. 204 – 224, 2006.

[28] D. R. Hush and B. G. Horne, "Progress in supervised neural networks," *IEEE Signal Processing Magazine*, vol. 10, no. 1, pp. 8–39, 1993.

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

[30] F. Ertam and G. Aydn, "Data classification with deep learning using tensorflow," in *Int. Conference on Computer Science and Engineering*, 2017, pp. 755–758.

[31] D. Valencia and A. Alimohammad, "An efficient hardware architecture for template matching-based spike sorting," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 3, pp. 481–492, 2019.

[32] Y. Suo *et al.*, "Energy-efficient multi-mode compressed sensing system for implantable neural recordings," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 5, pp. 0–0, 2014.

[33] V. Karkare, S. Gibson, D. Markovic, "A 130-μw, 64-channel neural spike-sorting DSP chip," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 5, pp. 1214–1222, 2011.

[34] ——, "A 75-μw, 16-channel neural spike-sorting processor with unsupervised clustering," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 9, pp. 2230–2238, 2013.

[35] M. Zamani, D. Jiang, and A. Demosthenous, "An adaptive neural spike processor with embedded active learning for improved unsupervised sorting accuracy," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 3, pp. 665–676, June 2018.

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

[37] H. Xu et al., "Unsupervised and real-time spike sorting chip for neural signal processing in hippocampal prosthesis," *Journal of neuroscience methods*, vol. 311, pp. 111–121, 2019.

[38] A. T. Do, et al., "An area-efficient 128-channel spike sorting processor for real-time neural recording with 0.175 μW/channel in 65-nm CMOS," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 1, pp. 126–137, 2019.

[39] J. Thies and A. Alimohammad, "Compact and low-power neural spike compression using undercomplete autoencoders," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2019.

[40] T. E. Motassadeq, "CCS vs NLDM comparison based on a complete automated correlation flow between primetime and HSPICE," in *Saudi International Electronics, Communications and Photonics Conference*, 2011, pp. 1–5.

[41] G. Bi and M. Poo, "Synaptic modification by correlated activity: Hebb's postulate revisited," *Annual Review of Neuroscience*, vol. 24, no. 1, pp. 139–166, 2001.

[42] Y. Gao et al., "Low-power ultrawideband wireless telemetry transceiver for medical sensor applications," *IEEE Trans. on Biomedical Engineering*, vol. 58, no. 3, pp. 768–772, 2011.

[43] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, 2012.

[44] S. Kim, P. Tathireddy, R. A. Normann, and F. Solzbacher, "Thermal impact of an active 3-d microelectrode array implanted in the brain," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 4, pp. 493–501, 2007.