Neural Spike Sorting using Binarized Neural Networks

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

***Abstract*—This article presents the design and efficient hardware implementation of binarized neural networks (BNNs) for brain-implantable neural spike sorting. In contrast to the conventional artificial neural networks (ANNs), in which the weights and activation functions of neurons are represented using real values, the BNNs utilize binarized weights and activation functions to dramatically reduce the memory requirement and computational complexity of the ANNs. The designed BNN is trained using several realistic neural datasets to verify its accuracy for neural spike sorting. The application-specific integrated circuit (ASIC) implementation of the designed BNN in a standard 0.18-μm CMOS process occupies 0.33 mm^2 of silicon area. Power consumption estimation of the ASIC layout shows that the BNN dissipates 2.02μW of power from a 1.8 V supply while operating at 24 kHz. The designed BNN-based spike sorting system is also implemented on a field-programmable gate array and is shown to reduce the required on-chip memory by 89% compared to those of the alternative state-of-the-art spike sorting systems. To the best of our knowledge, this is the first work employing BNNs for real-time *in vivo* neural spike sorting.**

I. Introduction

The human brain is composed of billions of neurons, communicating with one another via electro-chemical processes. The brain is responsible for nearly all functional behavior of the human body, from controlling the movement of limbs to regulating one's breathing. Neuro-degenerative diseases and spinal cord injuries can impede the brain's natural ability to communicate with the limbs and/or organs and hence, imposing severe limitations on patients' bodily functions. Neuroscientists are keen on studying the behavior of neurons, how their electro-chemical activities are correlated with one another, and exploring how to form alternative neural pathways on a patient's brain.

The brain's neurons fire action potentials, also known as spikes, when their cumulative input activity exceeds some threshold. The brain can then decode the spiking activity to control the movement of a limb or muscles responsible for respiration. One approach to overcome the degeneration of neural connectivity is to use brain-machine interfaces (BMIs) to form alternative pathways for neural signals. In recordings preformed by large and dense multi-electrode arrays (MEAs), an electrode may record combined signals from multiple neurons nearby and background noise. Spike sorting is the process of separating the electrical activity of individual neurons from noisy recorded signals [1]. Conventionally, spike sorting is performed offline using a computer or a machine (i.e., in-silico). The recorded neural signal is first filtered between 300 and 3000 Hz to remove background noise. Spikes are then detected from ambient neural noise. Specific features of the spike waveforms are then extracted and used as inputs to a clustering algorithm, which groups or classifies similar spikes belonging to specific neurons.

The hardware realization of accurate spike sorting systems allows neuro-scientists to examine the spiking behavior of individual neurons. Recent hardware implementations of spike sorting aim to perform the required neural signal processing algorithms either as a supervised process, which requires some level of initial offline processing [2], or as a fully automated unsupervised process [3] in real-time [2]–[12]. The designs which require pre-processing aim to reduce the computational complexity and hence, silicon area and power consumption significantly, by performing a set of algorithms offline on a computer to estimate the design parameters for the real-time operation of the *in vivo* spike sorting system. To avoid heat-related tissue damage, the brain-implantable device is commonly realized using extremely low-power application-specific integrated circuits (ASICs) [13]. Computational acceleration of spike sorting using field-programmable gate arrays (FPGAs) have also been reported [2], [3], [14]–[16].

In this article we present an efficient design and implementation of binarized neural networks (BNNs) for real-time *in vivo* classification of neural spikes. In contrast to the conventional artificial neural networks (ANNs), in which the weights and activation functions are represented using real values, the BNNs utilize binarized weights and activation functions to significantly reduce the memory requirements and computational complexity of the conventional ANNs [17]. The learning process of the employed BNN is performed offline, which entails a partially supervised spike sorting. Most neural signal processing systems involve some offline processing [2], [3], [5], [9], [18], [19], which results in a more area- and power-efficient realization. While a trained BNN offers classification for waveforms similar to those seen during initial training, the BNNs can also offer generalization for spike sorting using alternative techniques such as noise injection [20] and Dropout [21], which act as a form of regularization and leads to low overfitting and greater generalization. This article introduces the application of BNNs for spike sorting in an effort to significantly reduce the computational complexity and memory requirement of the system, while utilizing an initial off-line parameter estimation.

The rest of this article is organized as follows. Section II briefly describes the spike sorting process and the employed neural signal processing algorithms. Section III discusses the fundamental operations of the BNNs and, for the first time, introduces their application for real-time *in vivo* spike sorting. Section IV presents the hardware architecture design of the BNNs. The ASIC and FPGA characteristics and implementation results of the designed BNN-based spike sorting system are presented and compared with those of the state-of-the-

art realizations. The feasibility of the brain-implantable BNN-based spike sorting system for real-time processing of neural signals is also discussed. Finally, Section V makes some concluding remarks.

II. Spike Sorting

Fig. 1 shows the system-level block diagram of a BMI. BMIs translate (decode) neural signals recorded with a MEA [22] into commands for the direction of machines and a variety of prosthetic devices and hence, restoring impaired neural signal pathways. In conventional spike sorting systems, the spiking activity of neurons is first detected from the recorded, amplified, and filtered neural signals combined with ambient noise. Spike detection is usually done in two steps, pre-emphasis and thresholding. Pre-emphasis involves specific signal processing on the neural signals, such as computing the absolute value [23], the non-linear energy operator (NEO) [24], or the discrete wavelet transform (DWT) [25]. The thresholding step then compares the neural signal, or the pre-emphasized signal, to a threshold value. A spike is detected when the neural signal crosses an assumed threshold.

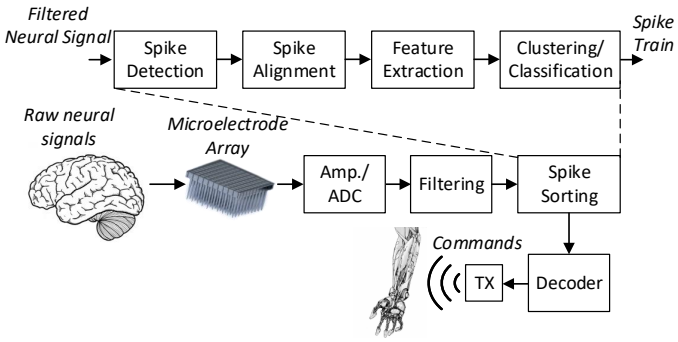

Fig. 1: The system-level diagram of a BCI.

After spike detection, spike alignment is performed such that each detected spike has a common reference point (e.g., maximum amplitude or maximum energy) at the same sample number in the waveform [1]. For example, if a spike waveform is represented using 64 samples, the alignment index indicates which of the 64 samples holds the alignment feature. Some commonly employed alignment features include the maximum slope, maximum absolute value, or maximum value of the spike waveform. In the alignment process, the distance between waveforms is commonly computed with the L_1-norm (Manhattan distance) or L_2-norm (Euclidean distance). The reliability of the distance computation can be improved by ensuring that the alignment feature is at the same sample index for all spikes. After spike alignment, feature extraction (FE) is utilized to reduce the dimensionality of the spike waveforms so that the characteristics of a spike waveform can be described using only a relatively small number of features. If FE is not used and the spikes are clustered directly, for example in a 64-dimensional space, a relatively large memory is required. However, it is far more efficient to interpret a two- or three-dimensional clustering space when representing the spike waveforms using a relatively small number of features, e.g., two or three, respectively.

The final step in the conventional spike sorting process is to group the extracted features into disjoint clusters. Feature extraction, clustering, and classification are closely tied together. The extracted features should be chosen such that the detected spikes originating from the same neuron create an individual cluster. Regardless of a particular FE approach, the classification process involves clustering of the features such that distinct features (and spike waveforms) create different groups (clusters). Early clustering methods involved manually identifying the clusters after extracting and plotting the features [1]. Two of the most commonly used clustering methods are the k-means [26] and OSort [27] algorithms. The k-means clustering algorithm begins by randomly defining k different cluster centroids. The data-points nearest to each cluster centroids are assigned to that cluster and the cluster centroid for each cluster is updated as the average of the cluster. One disadvantage of the k-means clustering algorithm, similar to the PCA algorithm, is that it cannot operate in real-time because it requires a relatively large number of data points to find the optimal cluster centroids. Even though recent improvements to the k-means algorithm involve computing relatively accurate approximations of the cluster centroids using a small number of data samples [28], the memory and computational requirements of processing high-dimensional data, such as neural spike waveforms, are still not adequate for area- and power-constrained brain-implantable ASICs. Also, the number of clusters (NOCs) k is not known apriori and requires supervision. In contrast, Osort creates the clusters on the fly by means of comparing the new input feature vector to all existing cluster averages. If the distance of the new feature vector is beyond the assignment threshold, a new cluster is created for that feature vector. The clustering algorithm associates spike waveforms with the corresponding cluster that the feature vector is assigned to. The OSort approach is thus attractive for hardware realization [3] because it is both unsupervised (i.e., does not require pre-processing for the estimation of cluster averages/centroids) and can cluster the spikes in real-time as spike waveforms are detected and their features are extracted.

III. Binarized Neural Networks for Spike Sorting

ANNs mimic brain neurons' activities with two key distinctions to traditional Von Neumann architectures. Firstly, the computation is performed by a set of processing elements, also referred to as artificial neurons (ANs), interconnected by weighted synapses. The processing elements are loosely modeled after biological neurons, where the excitation of a neuron depends on the activity of its pre-synaptic neurons. Secondly, the synaptic weights among neurons are parameters that are obtained via a training process. Training an ANN mimics how a human brain learns by example. An ANN consists of an input layer, zero or more hidden layers, and one output layer with various number of ANs in each layer. For each AN in the hidden and output layers, the weighted outputs from pre-synaptic neurons are accumulated and a non-linear activation function is applied to the result. Commonly employed non-linear activation functions include the sigmoid

function $f_{\mathrm{sigmoid}}(z) = \frac{1}{1+e^{-z}}$, the hyperbolic tangent function $f_{\tanh}(z) = \frac{2}{1+e^{-2z}-1}$, and the rectified linear unit $f_{\mathrm{ReLU}}(z) = z$ if $z \geq 0$, else $z = 0$, where z denotes the accumulated weighted inputs to an AN [29]. Each AN may have a bias b, which essentially adds a lateral shift to the activation function and effectively tunes how easily or difficult it is to produce an excited output. The parameters of a neural network, including weights, biases, the number of hidden layers, and the number of neurons in each hidden layer, are adjusted during training. The numerical resolution of the weights and biases, as well as the number of neurons in the hidden and output layers, pose a constraint on the overall memory requirement of the ANN. The impacts of reducing the parameter resolution on the training and performance of ANNs have been studied in [30], [31].

Deep neural networks (DNNs) employ a relatively large number of hidden layers and have been used in a variety of machine learning applications, from image and pattern recognition, self-driving vehicles, to medical experimentations for diagnosing diseases. The challenge with the efficient hardware implementation of DNNs for extremely area- and power-constrained applications, such as brain-implantable devices, lies in the relatively large number of parameters for storage. For off-line realizations, this constraint is not crucial due to the availability of storage on conventional workstations.

We have previously employed ANNs for spike sorting [4]. This work aims to further reduce the overall logical resource and memory requirements of the implemented spike sorting system. In a notable effort to reduce the memory requirement of DNNs, a method for the binarization of weights and activation functions was presented in [17]. The binarization is given as:

$$k_b = \mathrm{sign}(k) = \begin{cases} +1 \text{ if } k \geq 0, \\ -1 \text{ if } k < 0, \end{cases} \tag{1}$$

where k_b denotes the binarized value of k and sign denotes the sign function. As opposed to the traditional ANNs that often employ biases in the activation function, BNNs do not add a bias to the weighted inputs since it will cancel the binarization. The binarization of the learned parameters and the activation functions dramatically reduces the amount of memory required for BNNs. For example, consider a network topology of 784–512–10 with 784 input neurons, 512 neurons in the hidden layer, and 10 output neurons. The number of weight parameters for this topology is 406,528. If the weight values for a traditional ANN are stored using 8 bits, the total memory requirement is 3.25 MB, whereas the memory requirement for a BNN with the same network topology is only 406 kbits. Even if the weights are stored using two-bit signed integers, i.e., 11_2 and 01_2 for -1 and +1, respectively, the memory requirement is 812 kbits, which is far less than that of the ANN. Moreover, the required storage for representing the neurons' outputs is considerably smaller for the BNNs.

The accuracy of various BNNs have been assessed using standard datasets, including MNIST, SVHN, and CIFAR-10, and it was shown that the accuracy of the BNNs is comparable with those of the state-of-the-art classifiers [32]. To train the BNNs for spike sorting, we use the datasets in [33] and quantify the accuracy of the sorting process. The datasets consist of simulated waveforms which differ based on the differentiality of the spike shapes (*Easy* and *Difficult*) for various noise levels. Each dataset offers three distinct spike shapes. The spike shapes were randomly chosen from 594 spike shapes in a real neural signal database. Then, background noise was added to the spike waveforms with the noise standard deviation levels within [0.05, 0.2], and up to 0.4 for the Easy1 dataset, relative to the spike amplitude. The signal-to-noise ratios (SNRs) of the datasets are between -31.69 dB and -27.39 dB. Each dataset includes various ground truth information, such as the time of spike events, which indicates the particular spike shape (or spike class) that occurred at that time. The input to the BNN is a 64-sample aligned spike waveform and the three one-bit outputs of the BNN indicate the spike class that the input spike belongs to. For the binarized outputs of the network, two output encodings of one-hot and binary are utilized. The employed BNN topology is 64–5–n, i.e., 64 inputs, 5 neurons in a hidden layer, and n neurons in the output layer, where n is either equal to C for one-hot encoding or $\mathrm{ceil}\big(\log_2 C\big)$ for the binary encoding, where C denotes the number of spike classes (three in our designed and implemented BNN classifier) and ceil denotes the ceiling operator.

The Python Larq framework [34] is used to train the BNNs. The Adam optimizer with the default parameter values of learning rate = 0.01, β_1 = 0.9, β_2 = 0.999, ϵ = 1e-07, is used to train the BNNs for 250 epochs. For the dataset in [33], the training data consists of 2460 spikes and their corresponding spike classes. The data is divided into 70% for training and 30% for testing. The data is distributed such that each of the spike classes in the dataset appears in equal numbers and hence, the network is trained using a balanced dataset. The spike classes, which are given as integers in the original dataset, are converted into row vectors indicating the target outputs for the given training spikes with the desired output encoding mode. For example, if the spike class is 3, the one-hot encoded row vector denotes 001_2, i.e., the third output neuron should produce a one. Similarly, the binary encoded output is given as 11_2. Because the sign function only results in outputs -1 and 1, the zeros are mapped to -1. So, a spike class of 3 is mapped onto row vectors [-1 -1 1] and [1 1] for the one-hot and binary encoding, respectively.

Figs. 2(a–d) show the classification accuracy of the trained BNN using the one-hot and binary encoding schemes, over the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. One can see that for most cases, the binary output encoding outperforms the one-hot encoding. The classification accuracy degrades with increasing the noise level for both encoding schemes, however, utilizing the binary encoding, the accuracy of the BNN is less susceptible to increases in the noise level over all the datasets compared to utilizing one-hot encoding. The median classification accuracy of the one-hot encoded BNNs are 0.96, 0.91, 0.78, and 0.78, while for the binary encoded BNN are 0.98 0.96, 0.89, and 0.82, for the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. Table I gives the average classification accuracy of the BNN employ-

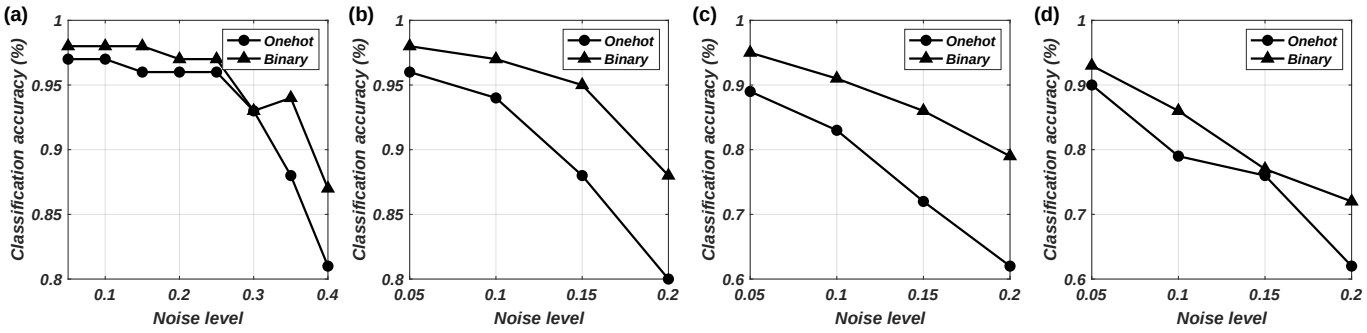

Fig. 2: Classification accuracy of the BNN for the (a) Easy1, (b) Easy2, (c) Difficult1, and (d) Difficult2 datasets using the one-hot and binary encoding schemes.

ing two encoding schemes over four different datasets and various noise levels. One can see that for the employed datasets, the binary encoding provides greater average accuracy.

TABLE I: The average classification accuracy of the BNN employing two encoding schemes over four different datasets and various noise levels.

Dataset	Encoding method	Classification accuracy (%)
C_Easy1	One-hot	92
	Binary	95.25
C_Easy2	One-hot	89.5
	Binary	94.5
C_Diff1	One-hot	76.5
	Binary	87.75
C_Diff2	One-hot	76.75
	Binary	82

We also trained the BNN using two other datasets, a multi-unit activity dataset [35] and a real human dataset [36]. A multi-unit activity is defined as the aggregate response of multiple single-units near the tip of an electrode. The multi-unit dataset was created to more accurately model the realistic conditions of recorded neural signals using a relatively dense MEA. The background noise was modeled by the superposition of the spiking activity of extracellular neurons. The five defined multi-unit datasets have varying amplitudes for the single-unit activities and have different spike firing rates. For classification, the datasets have three possible spike shapes: two distinct single-unit waveforms and a set of multi-unit waveforms. The average median classification accuracy for the one-hot encoded BNN is 0.76 and for the binary encoded BNN is 0.81 over the multi-unit datasets 1 – 5. Again, the binary output encoding shows a better classification accuracy than the one-hot encoding. Because the binary encoding can cause output neurons to produce the same output value for different types of spike waveforms, this can be used as a form of regularization for the network during training.

The real human dataset is a recording from the temporal lobe of an epileptic patient. Because verified spike times and spike classes were not included in the real human dataset, we use the OSort algorithm to cluster a subset of the spike waveforms, which then converged to four distinct clusters. The cluster assignments were used to train the BNN using a subset of the spike waveforms. The BNN was then evaluated using the remainder of the spikes in the dataset. The classification accuracy of the BNN on the real human dataset was 0.72 and 0.88 using the one-hot encoding and the binary encoding schemes, respectively.

Some spike sorting systems employ feature extraction to reduce the dimensionality of the spike waveforms [5], [8], [9], [12]. To compare the sorting accuracy of the BNN-based classifier when using detected spike waveforms and a reduced feature space, we employed zero-crossing features (ZCFs) given as [37]:

$$\mathrm{ZC}_1 = \sum_{n=0}^{K_1-1} s(n), \qquad \mathrm{ZC}_2 = \sum_{n=C}^{K_2-1} s(n),$$

where K_1 denotes the zero-crossing point in the spike waveform and K_2 denotes the number of samples in the spike waveform $s(n)$. ZC_1 and ZC_2 denote the areas under the positive and negative portions of the spike waveform, respectively. They reduce the dimensionality of the spike feature space by 96%. The ZCF would require two additional accumulators to compute ZC_1 and ZC_2 and the network topology becomes 2–5–3 for the one-hot encoding and 2–5–2 for the binary encoding, resulting in the weight memory requirements of only 25 bits and 20 bits, respectively. The dimensionality reduction of the network input and the weight memory requirements comes at the significant loss in classification accuracy (around 65% and 36% for the binary and one-hot encoded outputs, respectively). It is also shown that increasing the number of hidden layers and the number of ANs in a layer yielded negligible performance improvement. Therefore, we do not incorporate feature extraction in our designed spike sorting system.

IV. BNN Hardware Architecture

Fig. 3 shows the top-level block diagram of the designed BNN-based spike sorting system using the NEO-based spike detector [2], the maximum amplitude spike alignment, and the BNN-based spike classifier. Because our goal is to reduce the silicon area and power consumption of the spike sorting circuit for real-time *in vivo* realization while achieving an acceptable level of sorting accuracy, we choose to use the NEO-based detection technique as it imposes less computational complexity than computing the DWT, yet providing improved

accuracy over the absolute value method of pre-emphasis. This is primarily because the NEO algorithm derives its threshold value based on the probability of false alarm, whereas the threshold value in the absolute method is not adaptive to real-time characteristics of the neural signal. It was shown in [38] that in cases where the signal-to-noise ratio (SNR) was relatively high, the absolute value method is as accurate as NEO for spike detection. However, NEO outperforms the absolute value method when the SNR is low.

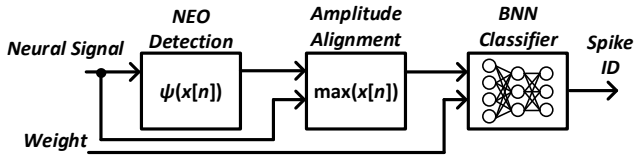

Fig. 3: The top-level block diagram of the BNN-based spike sorting hardware.

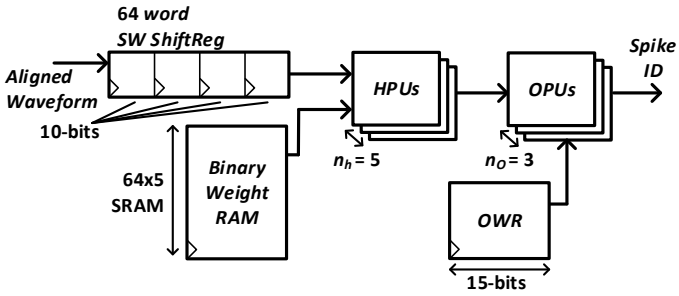

Fig. 4: The block diagram of the designed BNN-based spike classification module.

The block diagram of the BNN-based spike classification module is shown in Fig. 4. The aligned spike waveform is stored in the shift register *SW ShiftReg*. To accommodate the aligned spikes, the depth of *SW ShiftReg* is set to 64 and each of the registers is 10 bits wide. The main processing units are the hidden layer processing units (*HPUs*) and the output layer processing units (*OPUs*). Once the aligned spike is received, the control unit begins to read values from the weight memory *Binary Weight RAM* (*BWRAM*) into the *HPU*. To avoid using two bits to represent the values -1 and 1, a zero output of the sign function denotes a positive sign bit and correlates to a +1 output. Similarly, a one output of the sign function denotes a negative sign bit and correlates to a -1 output. Because the weights are constrained to -1 or +1 during the forward propagation, the accumulation of gradients is also constrained to the range between -1 and 1 during the training process. The width of the *BWRAM* is equal to the number of hidden layer nodes $n_H = 5$ and its depth is equal to the number of samples used to store the spike, i.e., 64. The *BWRAM* is implemented using the SRAM standard cells. The synaptic weights between the hidden layer and the output layer are stored in a 15-bit output weight register *OWR*. The number of bits stored in *OWR* is equal to the product of the number of hidden layer neurons n_h and the number of output layer neurons n_O. For our designed network topology with 64 inputs, 5 hidden layer neurons, and 3 output neurons, the synaptic weight memory is only 335 bits. Compared to our template matching (TM)-based spike classifier in [2], which requires 3072 bits of storage for three templates, the designed BNN-based spike classifier requires 89% less storage.

The block diagram of the *HPUs* is shown in Fig. 5. It consists of a two's complement unit, to negate the spike samples, and an accumulator. Since the spike inputs are real-valued and the weights associated with the synaptic connections may or may not change the signs of the spike samples, the hidden layer needs to accumulate non-binary input values. The weight values read from the *BWR* determine whether the value that is accumulated is the negated spike sample value or the original value for the weight 1 or 0, respectively. The control signal *AccRST* is used to reset the accumulator registers. The *WI.WF* denotes the integer and fraction lengths of the accumulation registers, respectively. The sign function is implemented by passing the most significant bit of the accumulating registers to the output. To process all n_h hidden layer neurons in one clock cycle, the *HPU* consists of n_h sets of multiplexers and accumulators, producing n_h bits at its output.

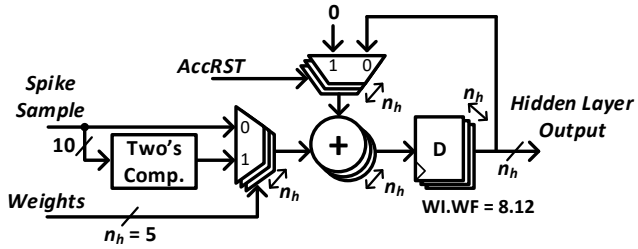

Fig. 5: The block diagram of the hidden layer processing units.

Fig. 6 shows the block diagram of the output layer processing units. It consists of an array of n_O XOR gates, majority functions $m(X)$, where X denotes the outputs of the XOR arrays, and a set of inverters. The XOR arrays are effectively used to perform the bitwise multiplication between the outputs of the *HPU* and the weight values stored in the *OWR*. The output of each XOR array is n_h bits wide and the majority function $m(X)$ is used to determine whether the n_h bits contain more ones than zeros. For our designed BNN, since $n_n = 5$, $m(x) = 1$ when at least three of the bits in X are one. The set of inverters is only used when the one-hot encoding is employed, in which case the value of each output bit corresponds to the presence or lack of a particular spike class at that time.

To assess the accuracy of the designed BNN classifier over varying number of clusters, we employed the datasets from [40], which contains 95 simulations of neural signals with three to twenty clusters. The classification accuracy in Fig. 7 shows that the designed BNN with only five ANs in the hidden layer can reliably sort the activity of up to 12 single units with the binary encoding scheme. When utilizing the binary encoding scheme, the NOC is limited to a maximum of $2^{n_o} - 1$, where n_o denotes the number of ANs in the output layer. Thus, the designed BNN-based spike sorting system with $n_o = 3$ can sort up to seven clusters. The memory size of the one-hot encoded BNN can be given as $(64 \times n_h) + (n_h \times NOC)$, where 64 is the number of samples

TABLE II: The ASIC characteristics and implementation results of various spike sorting systems

Design	Ours	[3]	[2]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
Algorithm	BNN	OSort	TM	ANN	FE	OSort	OSort	FE	FE	TM	FE	FE*
Number of clusters per channel	3–7	3	3	3	–	3	3	3	4	3	3	6
Classification accuracy	0.91–0.90	0.87	0.90	0.98	0.77	0.75	0.93	0.84	0.85	0.93	0.86	0.92
Data rate reduction	2000×	1600×	3200×	1866×	11×	240×	278×	240×	–	–	257×	–
Supervised	Y	N	Y	Y	Y	N	N	N	Y	N	N	N
Technology (nm)	180	32	45	32	90	65	45	45	130	40	65	65
Core voltage (V)	1.8	1.16	0.25	0.7	0.55	0.27	–	1.1	1.2	1.1	0.54	1
Operating frequency (kHz)	24	24	24	20	4000	480	56	960	160	500	3200	30
Area per channel (mm^2)	0.33	2.57	0.30	0.009	0.06	0.07	0.07	2.7	0.023	0.0175	0.003	0.09
Power per channel (μW)	2.02	2.78	0.064	1.04	2	4.68	10.3	20	0.75	19	0.175	0.181
Scaled area per channel (mm^2)*	0.33	102.8	5.7	0.36	0.396	0.84	1.33	51.3	0.066	0.473	0.036	1.08
Scaled power per channel (μW)*	2.02	8.849	6.04	15.04	56.89	671.61	–	57.99	1.86	56.8	6.28	1.06

* Scaled to a 180-nm CMOS technology with a 1.8V supply voltage, as defined in [39]. Our implementation results are highlighted.

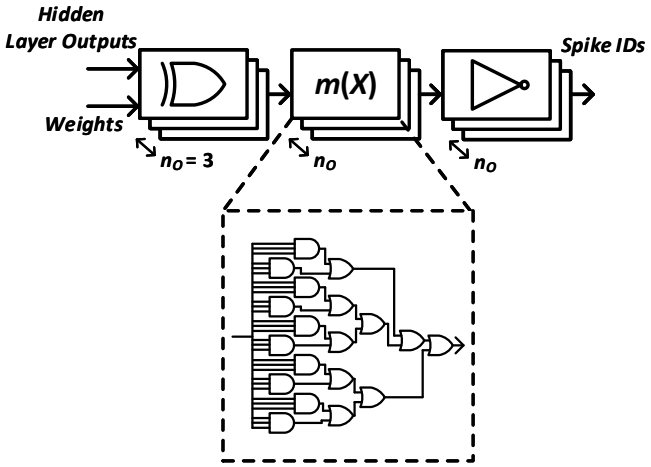

Fig. 6: The block diagram of the output layer processing units (*OPUs*).

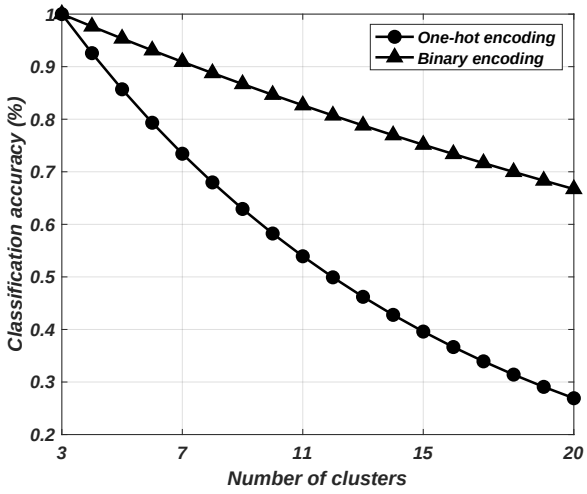

Fig. 7: Classification accuracy of the BNN utilizing the one-hot and binary encoding schemes for varying numbers of clusters.

in a spike waveform and n_h denotes the number of ANs in the hidden layer. The memory size of the binary encoded BNN

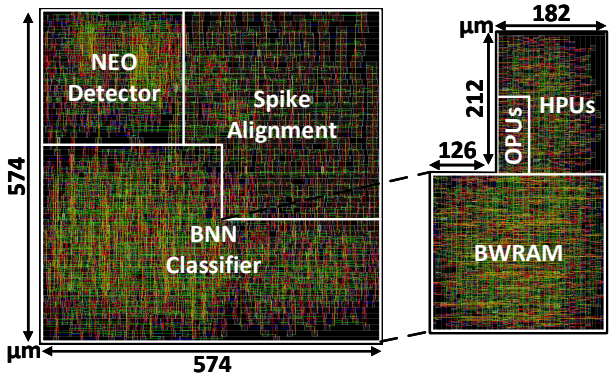

Fig. 8: The chip layout of the designed BNN-based spike sorting system.

can be given as $(64 \times n_h) + (n_h \times \text{ceil}[\log_2(NOC)])$. Because the binary-encoded BNN is able to represent a wider range of output values using fewer ANs in the output layer, it lowers the number of hidden-to-output layer weights. Moreover, the binary encoded BNN offers a greater accuracy for different number of clusters and over varying noise levels. Compared to the other state-of-the-art spike sorting systems, offering up to six clusters per channel, the designed BNN-based spike sorting system can classify up to 20 clusters.

The training time for large-scale and deep neural network models could be considerable, however we found that since our model is relatively small, increasing the NOCs, which increases the number of ANs in the output layer, has a relatively small impact on the training time. However, the training time for the estimation of the binary weights is directly related to the number of training examples, the number of training epochs, and the number of training iterations. The off-chip training time of our *in vivo* BNN-based spike sorting system ranges between 250 ms and 400 ms for 10 iterations of training with 50 epochs per iteration.

We have implemented our BNN-based spike sorting system in a standard 180-nm TSMC CMOS process. The chip layout of the BNN-based sorting system is shown in Fig. 8, which is estimated to occupy $0.33\ \text{mm}^2$ of silicon area. We previously designed and implemented the NEO-based spike detection

and spike alignment units for our TM-based spike sorting system and the parallel OSort-based real-time spike sorting system in [2] and [3], respectively. Synthesis was performed using Synopsys Design Compiler and the place-and-routing was performed using Cadence Innovus. After routing and static timing analysis, the power was estimated by simulating the layout with the datasets from [33]. A variable change dump file was used to more accurately estimate the switching activity in Innovus. It was estimated that the BNN-based spike sorting chip will consume 2.02 μW from a 1.8 V supply while operating at 24 kHz.

A commonly used metric for quantifying the classification accuracy of spike sorting realization is the F-score metric [41] and is given as $F = \frac{2T_P}{2T_P+F_P+F_N}$, where T_P, F_P, and F_N denote the number of true positive, false positive, and false negative classifications, respectively. True positives are defined as spikes that are detected, classified, and exist in the reference dataset. A true positive is verified by finding the spike waveform in the reference dataset as well as the time of detection. A false positive is a spike that is detected and classified, but does not exist in the reference dataset. A false negative is a spike that exists in the reference dataset, but is not classified by the spike sorting system. Our ASIC implementation of the BNN-based spike sorting system achieve the average F-Scores of 0.96, 0.94, 0.86, and 0.86 for the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. The F-scores for the multi-unit datasets 1–5 are given as 0.94, 0.83, 0.89, 0.84, and 0.96, respectively. The F-scores for the datasets from [40] with three to seven NOCs are given as 1.0, 0.98, 0.97, 0.96, and 0.95, respectively. The F-score cannot be computed for the real human dataset due to the lack of the ground truth. The designed ASIC classifier is a bit-true realization of the fixed-point software model and hence, their F-scores are equivalent.

Various ASIC implementations of spike sorting systems have been previously reported [2]–[11]. Table II gives the characteristics and implementation results of various state-of-the-art spike sorting systems. Our designed and implemented BNN-based spike sorting system is able to classify single units of up to seven clusters. In [3] we designed and implemented a parallel OSort-based spike sorting system in a standard 32-nm CMOS process. In [2] we implemented the TM-based spike sorting ASIC using three templates in a 45-nm CMOS process. In [4] we implemented an ANN-based spike sorting system. The ANN consists of one hidden layer neuron and three output layer neurons, all of which utilize the ReLU activation function. The work in [5] performs NEO spike detection, aligns detected spikes to maximum derivative, and implements FE via discrete derivatives. Their design consists of four 16-channel modules which produce either aligned spikes or feature vectors, but does not perform real-time classification of the detected spikes. The work in [6] uses the absolute value detection scheme and implements the OSort clustering algorithm for a single 16-channel module. Similar to [6], in [7] spikes are first detected using a voltage threshold. Detected spikes are then aligned to a maximum absolute amplitude and OSort-based clustering was utilized. The work in [8] performs single-channel spike sorting using the NEO-based detection, maximum amplitude alignment, and FE using discrete derivatives. It supports an unsupervised learning process, similar to the OSort-based systems. The work in [9] presents a multi-channel spike sorting ASIC based on FE. The design in [10] presents a multi-channel TM-based spike sorting ASIC with a built-in OSort learning system. The work in [11] presents a multi-channel spike sorting processor based on the integer coefficient FE and clustering. Finally, the work in [12] presents the ASIC implementation of a dictionary learning-based feature extraction. Similar to the BNN approach, the dictionary values are constrained to the ternary set of [-1, 0, +1] and no multiplications are required. Note that the authors only list the ASIC implementation results for the feature extraction module, which are denoted as "FE*" in Table II. The spike sorting systems, which employ unsupervised learning, support real-time clustering/classification of detected spike waveforms. However, the TM-based and BNN-based sorting systems that require pre-processing of the neural recording in order to generate the template waveforms and BNN weights, respectively, require significantly less storage and reduced computational requirements, while achieving comparable sorting accuracy. The ASIC designs in [5], [6], [9]–[11] are multi-channel systems and, for a fair comparison, the area and power consumption results for single-channel sorting systems are given in Table II. During spike sorting operation, the sampling rate is 24 kSamp/sec. Each sample is represented using 10 bits, which results in an input bitrate of 240 Kbps. With an average neuron spiking rate of 40 spikes per second [42] and representing the BNN output classification with 3 bits (one bit per neuron in the output layer), the output bit rate is reduced to 120 bps. This results in a 2000 times data rate reduction compared to the input sampling rate. Since the energy required to transmit one bit of data is approximately 3 nJ [43], this results in a wireless transmission power of about 360 nW. Therefore, the total power of our ASIC chip is 2.36 μW with the power density of 7.15 $\mu\mathrm{W}/\mathrm{mm}^2$, which satisfies the tissue-safe requirements for brain implantable devices [44].

For a fair comparison of the ASIC implementation results across different CMOS technologies and supply voltages, we have also reported the power consumption and area utilization of the previously published work scaled to 180-nm technology as presented in [39]. As given in Table II, the BNN-based spike sorting system requires a smaller silicon area and consumes less power, while providing comparable sorting accuracy. Our TM-based spike sorting ASIC [2] and the OSort-based clustering system [3], as well as the work in [7], [8], [10], [11], have utilized the same reference datasets from [33] to quantify the accuracy of their sorting system. The work in [5] uses both synthetic data and real data, but the synthetic data differs from the one used in [33]. The work in [6] and [9] both use real neural data for evaluating the accuracy of their spike sorting systems.

Compared to the OSort clustering, in which the algorithm can adapt to signal variations in real time, such as electrode drift or increases in noise levels, the BNN-based clustering does not offer run-time adaptability. Nevertheless, compared to the other supervised approaches that require pre-processing for parameter estimation, such as TM-based spike sorting,

TABLE III: The characteristics and implementation results of various spike sorting systems on FPGAs

Work	Algorithm	Device	WL.WF	Regs.	LUTs.	BRAMs	DSP48s.	Max. Freq. (MHz)	Clustering Latency	Sorting Accuracy
Ours	BNN	Artix-7	16.11	267 (0.1%)	314 (0.3%)	2 (0.27%)	0 (0%)	125	0.51 μs	93%
[3]	OSort	Virtex-6	16.11	8444 (2%)	16472 (9%)	29 (4%)	130 (9%)	123	0.26 μs	87%
[14]	OSort	Virtex-5	16.8	16245 (27%)	23567 (40%)	63 (25%)	29 (4%)	100	11.1 ms	–
[2]	TM	Virtex-6	16.11	4880 (1%)	6635 (3%)	0 (0%)	5 (0.6%)	122	0.55 μs	90%
[15]	TM	Virtex-6	20.0	29000 (6%)	190000 (83%)	24 (6%)	–	–	2.65 ms	–
Ours*	BNN	Zynq-7000	16.11	124 (0.05%)	147 (0.12%)	0 (0%)	0 (0%)	263	0.24 μs	–
[16]	OSort	Zynq-7000	16.0	12150 (11%)	14037 (16%)	102 (72%)	120 (54%)	101	179.4 μs	–

Our implementation results are highlighted.

the BNN can employ various optimization schemes during training to create a generalized and robust network model that reduces overfitting [45]. For example, the training dataset can be augmented by additional data with increased noise levels to improve the robustness of the BNN model [20]. An alternative approach to diminish overfitting is to introduce Dropout among layers of the BNN by removing some synapses during training and hence, forcing the network model to "space out" it's weighting of parameters [21].

Our BNN-based spike sorting system is portable and designed for FPGA and ASIC realizations. Both ASIC and FPGA implementations were tested and achieved the same sorting accuracy. Table III gives the characteristics and implementation results of various state-of-the-art spike sorting systems implemented on FPGAs. In Table III, for a fair comparison, we list the implementation results for the designed BNN-based spike sorting system using similar FPGA devices. One can see that the unsupervised Osort-based implementation requires considerably more reconfigurable resources than that of the TM-based and that of the supervised BNN-based sorting systems. The design in [14] presents an FPGA implementation of OSort, which was designed for high-performance processing of recorded neural data on a workstation. The reported latency of 11 ms was assumed based on a 24 kHz sampling frequency, which implies the worst case sorting latency of 266 clock cycles. The work in [15] presents the hardware implementation of a bayes-optimal TM-based spike sorting system. While the maximum operating frequency of their design was not reported, their sorting latency was given as 53 clock cycles, which is slightly less than that of our design at 64 clock cycles. Since the work in [16], which uses the OSort algorithm for real-time clustering of spikes, only presents the FPGA implementation results for the clustering and classification modules, for a fair comparison the FPGA implementation results of the designed BNN classification module are given separately and denoted by "Ours*" in Table III. Note that the classification latency of the OSort-based implementation in [16] is 18,127 clock cycles while the classification latency of our BNN-based classification module is only 64 clock cycles.

V. Conclusion

This article presented that the binarized neural networks (BNNs) can be efficiently used for real-time *in vivo* spike sorting, while significantly reducing the memory requirement and computational complexity of the system compared to the other state-of-the-art realizations. Our ASIC implementation results confirmed that a relatively small BNN can be used for brain-implantation due to its small silicon area and its low power consumption, while providing reliable sorting accuracy for varying numbers of clusters.

Acknowledgment

This work was supported by the NSF Award 2007131 and, in part, by the Center for Neurotechnology (CNT), a National Science Foundation Engineering Research Center (EEC-1028725).

References

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

[2] 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.

[3] ——, "A real-time spike sorting system using parallel osort clustering," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1700–1713, 2019.

[4] D. Valencia, J. Thies, and A. Alimohammad, "Frameworks for efficient brain-computer interfacing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1714–1722, 2019.

[5] V. Karkare, S. Gibson, and 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.

[6] ——, "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.

[7] Y. Liu, J. Sheng, and M. C. Herbordt, "A hardware design for in-brain neural spike sorting," in *IEEE High Performance Extreme Computing Conference*, 2016, pp. 1–6.

[8] 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, 2018.

[9] 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 Transactions on Biomedical Circuits and Systems*, vol. 11, no. 4, pp. 743–754, 2017.

[10] 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.

[11] A. 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 VLSI Systems*, vol. 27, no. 1, pp. 126–137, 2019.

[12] M. Zamani, J. Sokolić, D. Jiang, F. Renna, M. R. Rodrigues, and A. Demosthenous, "Accurate, very low computational complexity spike sorting using unsupervised matched subspace learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 2, pp. 221–231, 2020.

[13] H. Ando, K. Takizawa, T. Yoshida, K. Matsushita, M. Hirata, and T. Suzuki, "Wireless multichannel neural recording with a 128-mbps uwb transmitter for an implantable brain-machine interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 6, pp. 1068–1078, 2016.

[14] S. Gibson, J. W. Judy, and D. Marković, "An FPGA-based platform for accelerated offline spike sorting," *Journal of Neuroscience Methods*, vol. 215, no. 1, pp. 1–11, 2013.

[15] J. Dragas, D. Jäckel, A. Hierlemann, and F. Franke, "Complexity optimization and high-throughput low-latency hardware implementation of a multi-electrode spike-sorting algorithm," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 2, pp. 149–158, 2015.

[16] L. Schäffer, Z. Nagy, Z. Kineses, and R. Fiáth, "FPGA-based neural probe positioning to improve spike sorting with OSort algorithm," in *IEEE International Symposium on Circuits and Systems*, 2017, pp. 1–4.

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[18] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C. Poon, "Real-time FPGA-based multichannel spike sorting using hebbian eigenfilters," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 4, pp. 502 – 515, 2011.

[19] Z. Jiang, J. P. Cerqueira, S. Kim, Q. Wang, and M. Seok, "1.74-μw/ch, 95.3%-accurate spike-sorting hardware based on bayesian decision," in *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*. IEEE, 2016, pp. 1–2.

[20] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," in *Advances in Neural Information Processing Systems*, 2017, pp. 5109–5118.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] R. R. Harrison *et al.*, "A low-power integrated circuit for a wireless 100-electrode neural recording system," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 123–133, 2007.

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

[24] 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.

[25] K. H. Kim and 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.

[26] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297.

[27] U. Rutishauser, E. Schuman, and 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] P. O. Olukanmi, F. Nelwamondo, and T. Marwala, "k-means-lite: Real time clustering for large datasets," in *International Conference on Soft Computing & Machine Intelligence*. IEEE, 2018, pp. 54–59.

[29] D. Valencia, S. F. Fard, and A. Alimohammad, "An artificial neural network processor with a custom instruction set architecture for embedded applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020.

[30] S. Wu *et al.*, "Training and inference with integers in deep neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HJGXzmspb

[31] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[32] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

[33] R. 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.

[34] L. Geiger and P. Team, "Larq: An open-source library for training binarized neural networks," *Journal of Open Source Software*, vol. 5, no. 45, p. 1746, Jan. 2020. [Online]. Available: https://doi.org/10.21105/joss.01746

[35] J. Martinez, C. Pedreira, M. J. Ison, and R. Q. Quiroga, "Realistic simulation of extracellular recordings," *Journal of neuroscience methods*, vol. 184, no. 2, pp. 285–293, 2009.

[36] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried, "Invariant visual representation by single neurons in the human brain," *Nature*, vol. 435, no. 7045, pp. 1102–1107, 2005.

[37] M. K. Awais and J. M. Andrew, "On-chip feature extraction for spike sorting in high density implantable neural recording systems," in *Biomedical Circuits and Systems Conference*. IEEE, 2010, pp. 13–16.

[38] S. Gibson, J. W. Judy, and D. Markovic, "Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 5, pp. 469–478, 2010.

[39] A. Stillmaker, Z. Xiao, and B. Baas, "Toward more accurate scaling estimates of cmos circuits from 180 nm to 22 nm," *VLSI Computation Lab, ECE Department, University of California, Davis, Tech. Rep. ECE-VCL-2011-4*, vol. 4, p. m8, 2011.

[40] C. Pedreira, J. Martinez, M. J. Ison, and R. Q. Quiroga, "How many neurons can we see with current spike sorting algorithms?" *Journal of Neuroscience Methods*, vol. 211, no. 1, pp. 58–65, 2012.

[41] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[42] S. Gibson, "Neural spike sorting in hardware: From theory to practice," Ph.D. dissertation, University of California Los Angeles, 2012.

[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.

[45] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks," in *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*. IEEE, 2017, pp. 173–177.