Biomedical Engineering Letters (2023) 13:73–83
https://doi.org/10.1007/s13534-022-00255-7

ORIGINAL ARTICLE

Partially binarized neural networks for efficient spike sorting

Daniel Valencia[1,2] · **Amir Alimohammad**[1]

Received: 18 July 2022 / Revised: 18 November 2022 / Accepted: 2 December 2022 / Published online: 9 December 2022

Abstract
While brain-implantable neural spike sorting can be realized using efficient algorithms, the presence of noise may make it difficult to maintain high-peformance sorting using conventional techniques. In this article, we explore the use of partially binarized neural networks (PBNNs), to the best of our knowledge for the first time, for sorting of neural spike feature vectors. It is shown that compared to the waveform template-based methods, PBNNs offer robust spike sorting over various datasets and noise levels. The ASIC implementation of the PBNN-based spike sorting system in a standard 180-nm CMOS process is presented. The post place and route simulations results show that the synthesized PBNN consumes only 0.59 μW of power from a 1.8 V supply while operating at 24 kHz and occupies 0.15 mm^2 of silicon area. It is shown that the designed PBNN-based spike sorting system not only offers comparable accuracy to the state-of-the-art spike sorting systems over various noise levels and datasets, it also occupies a smaller silicon area and consumes less power and energy. This makes PBNNs a viable alternative towards the implementation of brain-implantable spike sorting systems.

Keywords Neural networks · Brain-computer interfaces · Spike sorting · Application-specific integrated circuits · Neural signal processing

1 Introduction

The ability to efficiently record and decode neural signals is of vital importance towards the rehabilitation of patients with various neurodegenerative diseases, including Alzheimer's and Parkinson's.

A brain-machine interface (BMI) translates neural activities into commands for controlling external devices. For example, in [1] neural activity associated with imagined handwriting is used to convert the brain's activity into text. Also, in [2], neural signals were used to enable patients to synthesize speech directly from their thoughts. The algorithms employed for accurate neural decoding typically process spike trains, which represent the action potentials (or spikes) of individual neurons over time [3].

For greater spatial resolution and decoding performance, neural activities are first recorded by a multi-electrode array (MEA), which consists of intracortical electrodes capable of recording neural spikes from a relatively large number of neurons, in the order of hundreds, simultaneously. The recorded raw signals are then amplified and filtered into the frequency bands of interest. While each electrode records voltage signals fired by a neuron, or single-unit activities (SUAs), it also records spikes from neighboring neurons, or multi-unit activities (MUAs). Spike sorting, which is the process of associating recorded spikes to individual neurons is of prime importance for reliable neural decoding [4]. Spike sorting can be viewed as a clustering process where action potentials fired from a particular neuron with similar waveforms are grouped together. Spike sorting is conventionally performed over four steps, as shown in Fig. 1: (i) spike detection, (ii) spike alignment, (iii) feature extraction, and (iv) clustering. Spike detection involves detecting the spiking activity of an ensemble of neurons from the background noise. Alignment is the process of ensuring that each detected spike waveform is aligned to a particular metric, such as the maximum amplitude. Feature extraction (FE) is optionally used to reduce the dimensionality of spike waveforms by describing them using a relatively small set of features. Finally, clustering involves grouping similar spike waveforms and creating disjoint clusters of spike features,

✉ Daniel Valencia
dlvalencia@sdsu.edu

[1] Department of Electrical and Computer Engineering, San Diego State University, San Diego, USA

[2] Department of Electrical and Computer Engineering, University of California, La Jolla, USA

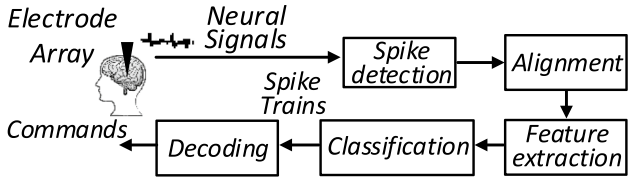

Fig. 1 The block diagram of a BMI system employing spike sorting

which identifies spikes originating from individual neurons. The clustering process also yields spike timings for each neuron that are subsequently used by the spike decoder to translate the generated spike trains into commands for controlling or communication with an external device [5].

While some designs implement spike sorting algorithms offline on a computer [6], some realizations based on template matching [7], neural networks [8, 9], and decision trees [10] employ offline estimation of parameters for in vivo sorting. Some techniques instead implement unsupervised learning algorithms in vivo to sort the detected spikes into clusters in real-time. For example, OSort creates and manages clusters as new spikes/features are made available [11–15], while *k*-means clustering iteratively updates cluster centroids using a set of training spikes/features. Due to advances in digital signal processing, the state-of-the-art brain-implantable spike sorting systems employ on-chip unsupervised learning, which inevitably leads to increased circuit area and power consumption. In our earlier work in [9], we reduced the power consumption and the silicon area of the spike sorting circuitry by designing a binarized neural network (BNN) for classifying spike waveforms using only bit-wise operations and employs an offline training for weight estimation. Unfortunately, the BNN-sorted spike waveforms are more susceptible to noise due to their higher dimension compared to the extracted features of spike waveforms. In this work, feature extraction is first applied to spike waveforms to reduce the dimensionality to only a few fundamental features. To improve classification performance compared to BNNs, we propose a partially binarized neural network (PBNN), which is less susceptible to noise while requiring fewer overall parameters for sorting. To the best of our knowledge, this is the first work employing a PBNN for neural network-based classification of feature vectors.

The rest of this article is organized as follows. Section 2 discusses the efficiency of various candidate feature extraction algorithms for hardware realization. Section 3 reviews the operation of PBNNs, presents their application in classifying extracted feature vectors, and discusses the accuracy of the proposed PBNN-based spike sorting system. The characteristics and implementation results of the designed spike sorting system are presented and compared with those of the state-of-the-art realizations in Sect. 4. Finally, Sect. 5 makes some concluding remarks.

2 Efficiency of the feature extraction methods

The principal component analysis (PCA) [4] and independent component analysis (ICA) [16] are two commonly employed feature extraction benchmark algorithms. Usually the first few principal components are the ones with the highest variations and hence, the dimensionality of a spike waveform can be reduced significantly. However, PCA has two fundamental limitations that hinders its feasibility for the in vivo implementations. Firstly, PCA requires a relatively large number of detected spikes to find the orthogonal basis vectors, which prevents its real-time realization. Secondly, the computational complexity of the PCA makes it impractical for extremely low-power brain-implantable devices [17]. Even though ICA outperforms PCA for signals with a relatively high noise level [18], ICA is also computationally-intensive and is unrealistic for in vivo implementations.

A number of more computationally-efficient feature extraction algorithms have been reported. For example, zero-crossing features (ZCFs) [19] are given as:

$$ZC_1 = \sum_{n=0}^{K_1-1} s[n], ZC_2 = \sum_{n=K_1}^{K_2-1} s[n],$$

where $s[n]$ denotes the detected spike waveform, K_1 and K_2 denote the zero-crossing point and number of samples in the waveform, respectively, and ZC_1 and ZC_2 denote the accumulated energy of the neural signal before and after the zero-crossing point, respectively. Similar to the zero-crossing features, the integral transform (IT) algorithm [20] has found applications in neural spike feature extraction. The IT features are given as:

$$I_A = \frac{1}{K_A} \sum_{n=n_A}^{n_A+K_A-1} s[n], I_B = \frac{1}{K_B} \sum_{n=n_B}^{n_B+K_B-1} s[n],$$

where n_A and n_B denote the first sample of the positive and negative phases, respectively, and K_A and K_B denote the total number of samples in the positive and negative phases, respectively. One notable weakness of the IT algorithm is that the indeces n_A and n_B are not known a priori and finding them requires analysis of the spike waveforms, which imposes additional latency and is undesirable for real-time spike sorting. The minimum delimitation (MD) feature

extraction algorithm [21] is relatively computationally-efficient and is given as:

$$MD_1 = \sum_{n=1}^{M} s[n], MD_2 = \sum_{n=M+1}^{K} s[n],$$

where M denotes the index position of the minimum value and K denotes the number of samples in the spike waveform. The ZCF, IT, and MD feature extraction algorithms commonly represent the spike waveforms using two features and hence, significantly reduce the memory requirement of the subsequent clustering module at the cost of a relatively small FE hardware. Some realizations may utilize more than only two features by including more details of the spike itself, such as the index of the minimum or maximum point.

Another class of the FE algorithms are based on the discrete derivatives (DD), which can be considered as a simplified discrete wavelet transform [22]. The DD algorithm computes the slope of the spike waveforms at different scaling factors δ and can be written as:

$$DD_{\delta}[n] = s[n] - s[n - \delta].$$

Various combinations of different scaling factors can be utilized for constructing the feature space. For different scaling factors, some key features of the spike waveform, such as the positive peaks, negative peaks, and peak-to-peak amplitudes are accentuated. For neural spike sorting, four variations of the DD algorithm have been studied: the maximum difference test (DD-MDT) [23], the first and second derivatives (DD-FS) [24], DD with uniform sampling (DD-US) [6], and DD with two-extrema sampling (DD-2Ex) [25]. The DD-MDT, which is considered a simplified version of the Lillefors Test [26], extracts the multimodal coefficients of each scaling factor. The DD-FS algorithm computes the first and second derivatives of the spike waveform to accentuate its geometric characteristics. Specifically, the first derivative can be used to interpret the variations of the spike waveform's gradient, while the second derivative emphasizes its low frequency characteristics. The DD-US algorithm involves computing the DD with three different scaling factors and downsampling of the DD waveforms at even intervals. The DD-2Ex algorithm creates the DD waveforms with two different scaling factors and extracts the minimum and maximum values for the two DD waveforms, representing a spike waveform using four features. Among the FE algorithms that are feasible for efficient online implementation, the notable algorithms are the MD, DD, and ZCF. It has been shown in [25] that DD-2Ex is a good candidate for feature extraction due to its immunity to noise and its tolerance for similar-shaped spike waveforms, the DD-2Ex with scaling factors $\delta = 3, 7$ performs the best based on its computational complexity, performance, and 16 times dimensionality reduction from $m = 64$ spike waveform samples to $n = 4$ features.

3 Partially binarized neural networks

To map the newly extracted spike's feature vector to a particular spike class in real-time, we propose to utilize a neural network-based classifier. The computation of neural networks is performed by a set of processing elements, so-called artificial neurons (ANs), which interact with one another through weighted connections (synapses). The synaptic weights control how the network responds to specific input stimuli. The accumulated weighted input activities, either from network input or from pre-synaptic ANs, is passed to a non-linear activation function to produce the AN's output [27].

3.1 Partially binarized neural networks

The size of networks and the numerical resolution of the synaptic weights and activation functions pose a strict limitation on the types of networks that can be used for extremely area- and power-constrained brain-implantable applications. Binarized neural networks (BNNs) have been introduced to significantly reduce the computational complexity and memory requirements by performing simple binary operations [28]. The binarization kernel is given as $k_b = sign(k)$, where k_b will be 1 for positive values of k and -1 otherwise.

It has been shown that using the binarization kernel at the input and output layers reduce the model's accuracy compared to binarizing only the hidden layers [29]. By performing feature extraction and representing the spike waveform with only 4 data points, it will be inevitably more difficult for the network to learn additional information from fewer data points. Moreover, as the spikes have already undergone feature extraction, binarizing the input layer of the BNN may further reduce the accuracy of the sorting process. In the proposed PBNN, the input layer's synaptic weights remain binarized, however, contrary to a BNN, the PBNN employs one of four different quantization modes: Mode 00, Mode 01, Mode 10, and Mode 11, which denote where the binarization kernel k_b is employed. Mode 00 does not employ k_b, Mode 01 applies it only to the output layer's weights, Mode 10 applies it only to the outputs from the previous layer, and Mode 11, as in a BNN, applies it to both the previous layer's outputs and the output layer's weights. For modes other than 11, we apply the sigmoid activation function to the network output. For hardware implementation, the complexity is reduced by using a quantized sigmoid function QSigmoid(z, b) that

Table 1 The classification accuracy of the PBNN over various datasets and quantization modes

Dataset	Mode 00	Mode 01	Mode 10	Mode 11	*k*-Med	Dataset	Mode 00	Mode 01	Mode 10	Mode 11	*k*-Med
Easy1 0.05	93.88	93.17	56.54	33.49	93.59	Easy2 0.15	90.11	60.61	31.039	34.69	88.43
Easy1 0.10	95.46	93.26	63.82	52.69	95.46	Easy2 0.20	82.93	61.1	36.048	32.57	80.31
Easy1 0.15	94.25	67.09	51.79	30.17	93.67	Difficult1 0.05	92.76	85.81	36.041	33.53	92.9
Easy1 0.20	93.23	76.33	56.90	35.10	93.09	Difficult1 0.10	91.95	72.02	32.318	33.33	91.01
Easy1 0.25	92.04	68.56	56.81	43.71	91.81	Difficult1 0.15	83.38	63.81	36.834	35.97	81.72
Easy1 0.30	87.55	67.69	53.02	40.79	88.77	Difficult1 0.20	72.25	61.93	38.506	31.62	68.81
Easy1 0.35	83.87	72.4	49.08	34.79	85.71	Difficult2 0.05	93.90	67.75	33.135	34.91	94.20
Easy1 0.40	81.78	63.71	50.81	35.39	79.0	Difficult2 0.10	90.62	64.93	49.206	34.92	90.04
Easy2 0.05	95.89	62.39	31.96	32.40	95.74	Difficult2 0.15	80.88	64.24	64.82	32.26	78.77
Easy2 0.10	93.11	61.43	36.36	33.87	93.18	Difficult2 0.20	72.10	65.52	57.36	34.7	72.67

will be equal to 1 if the accumulated weighted activity z is greater than or equal to the neuron's bias b. Because the sigmoid function is symmetrical around the y-axis at input $z = 0$ with $f_{\text{sigmoid}}(0) = 0.5$, QSigmoid($z$, b) can be considered as applying a threshold of 0.5 to the output of the sigmoid after adding the bias b. Note that the Qsigmoid activation function is only applied during the forward propagation of signals at the output layer during PBNN inference. Employing this approximation during the training phase makes it challenging to generate useful gradients for approximating the optimal parameters of the network.

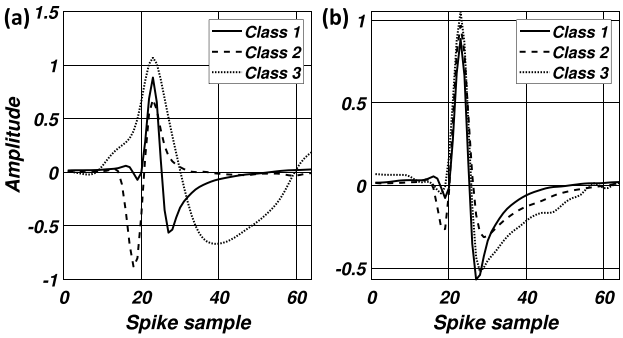

Fig. 2 The mean spike waveforms for the **a** Easy1 and **b** Difficult1 WaveClus datasets

3.2 Evaluation of BNN and PBNN for feature vector classification

We employ a PBNN as a feature vector classifier in our designed spike sorting system. In order to evaluate the performance of the spike sorting system, we employ the widely used WaveClus datasets [30]. The WaveClus datasets consist of 20 simulations of neural recordings with four levels of difficulty, Easy1, Easy2, Difficult1, and Difficult2. The difficulty level denotes the similarity of the spike waveforms between three single units present in the recording. The simulated recordings have varying levels of noise with a standard deviation between 0.05 and 0.40 relative to the amplitude of the spikes.

Figure 2**a** and **b** show the similarity of the spike waveforms for different classes in the Easy1 and Difficult1 datasets, respectively. One can see that the Difficult1 classes are harder to distinguish compared to the Easy1 dataset.

First, DD-2Ex feature vectors are extracted from detected spike waveforms. The employed network consists of four input units, one per feature, three hidden layer neurons, and three output layer neurons, one per spike class. The chosen network topology of 4–3–3 provided the highest classification accuracy over different datasets and noise levels and increasing the number and size of the hidden layers yielded a negligible increase in classification accuracy. The 4–3–3 network topology allows the PBNN to sort the spikes of up to three neurons using a one-hot encoded output. Clustering was performed first offline on a subset of the feature vectors using the *k*-medoids algorithm. The resulting clusters were then assigned identifiers 1–3, matching the spike classes present in the dataset. The PBNN is then trained to learn the mapping between feature vectors and spike classes defined by the *k*-medoids algorithm. Because the PBNN learns the mappings produced by the *k*-medoids algorithms, the performance of the clustering algorithm should be verified prior to training. After the initial clustering, the PBNN was trained on feature vectors extracted from the spike waveforms given in the WaveClus dataset using the Python Larq framework [31] extension for Tensorflow.

For optimizing the weight and bias parameters, we employ the RMSProp algorithm and a modified version of L2 regularization [32], which encourages the binarized weights toward values of − 1 and + 1. Layers of a PBNN that are not fully binarized employ normal L2 regularization rather than the modified L2. We train for 250 epochs, and use early stopping on the validation loss (mean squared error) to prevent overfitting of the model to the training set. Additionally, we utilize ten-fold

cross-validation to estimate the performance of the model accurately. The data is split in ratios of 80%, 10%, 10% for training, validation, and testing sets, respectively. For example, the Easy_0.10 dataset consists of 3522 spike waveforms, where the training, validation, and testing subsets are comprised of 2818, 352, and 352 feature vectors, respectively.

Table 1 gives the classification accuracy of the neural network utilizing the WaveClus datasets and four quantization modes. The *k*-Med column indicates the performance of the *k*-medoids algorithm by assigning the test subset features to the nearest cluster centroid. It is apparent that Mode 00 consistently outperforms other quantization modes. Another commonly employed metric for quantifying the performance of a classifier is the F-Score $F = \frac{2TP}{2TP+FP+FN}$, where TP denotes the number of true positive classifications, FP denotes the number of false positive classifications, and FN denotes the number of false negative classifications. Over all Wave_Clus datasets and noise levels, our model achieves a median F-Score of 0.91, ranging from 0.95 to 0.71 with the standard deviation of 0.072. The median F-Score is computed using the testing data subsets, which the model has not observed during training, for each of the ten cross-validation sets. Using the *k*-medoids clustering and the trained network, the performance of the BNN classifying spike waveforms is verified over the Wave_Clus datasets. The median classification accuracy over the low SNR datasets Easy1 0.30, Easy1 0.35, and Easy1 0.40 was 0.86, 0.78, and 0.67, respectively. Based on the results given in Table 1, the PBNN classification of feature vectors offers more robust accuracy for low SNR data.

While the weight values for Mode 00 were not constrained to + 1 or − 1 with the binarization kernel, we found that representing the output layer's weights with only 4 bits (2 bits for each of the integer and fractional parts) has a negligible impact of less than 0.9% on the classification accuracy. This implies the robustness of the PBNN for classifying feature vectors compared to the classification of spike waveforms using a BNN. Moreover, when classifying feature vectors, applying the binarization kernel to a layer's output imposes a greater degradation than binarizing only the weight values. For example, one can note a significant performance degradation from Mode 00 to Mode 10. Thus, when sorting feature vectors, we employ the Mode 00 PBNN over the BNN (Mode 11). Note that all following analyses are performed using a PBNN in Mode 00.

While the results in Table 1 show that the PBNN performs well for classifying the activity of three neurons, it is often not known how many neurons' spikes are present in the neural signal apriori. To study the effect of MUAs, the PBNN is trained to sort spikes of up to 20 different neurons [33]. The dataset of synthetic spikes is generated similarly to the WaveClus dataset, but provides information about the number of neurons that can be distinguished. Figure 3 shows the classification accuracy of the PBNN for increasing number of neurons in the neural recording. It can be seen that the PBNN can provide relatively accurate sorting for up to 7 units with 73% accuracy. Beyond this, it may be beneficial to increase the number of hidden units to allow the network more degrees of freedom for mapping features to cluster.

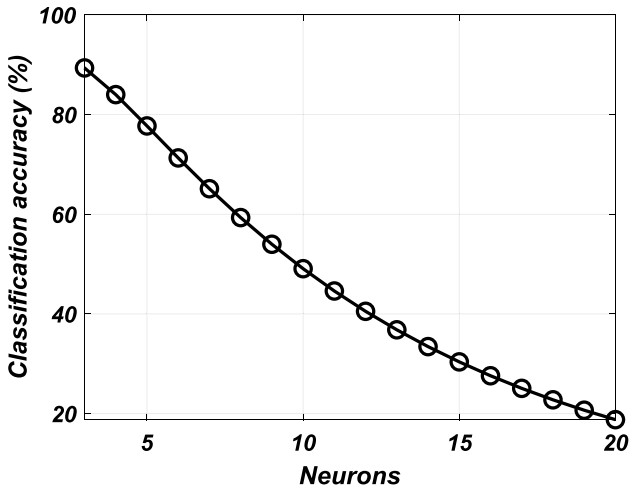

Fig. 3 The classification accuracy of the PBNN over varying number of neurons

3.3 Evaluation of BNN and PBNN-based sorting over real datasets

The ground truth information of the synthetic datasets offers a common benchmark for various sorting algorithms. Unfortunately, real neural recordings do not offer such ground truth information. We employ a two step approach for evaluating the performance of the spike sorting systems using real recorded data. Following spike detection, DD-2Ex features are extracted and a subset of them (80%) are clustered using the *k*-means algorithm. Since no ground truth information is available, we use the *k*-means cluster centroids as reference clusters, and assess the BNN and PBNN's performance by how well these networks classify spikes to match to their reference clusters. As given in Table 1, the conventional BNN does not perform well for classifying feature vectors.

To evaluate the BNN- and PBNN-based sorting schemes, we use a total of four neural recordings from two pigtail macaques (two each), aliases J and K, at the Washington National Primate Research Center. For training the BNN and PBNN, the same approach described in Sect. 3.B is

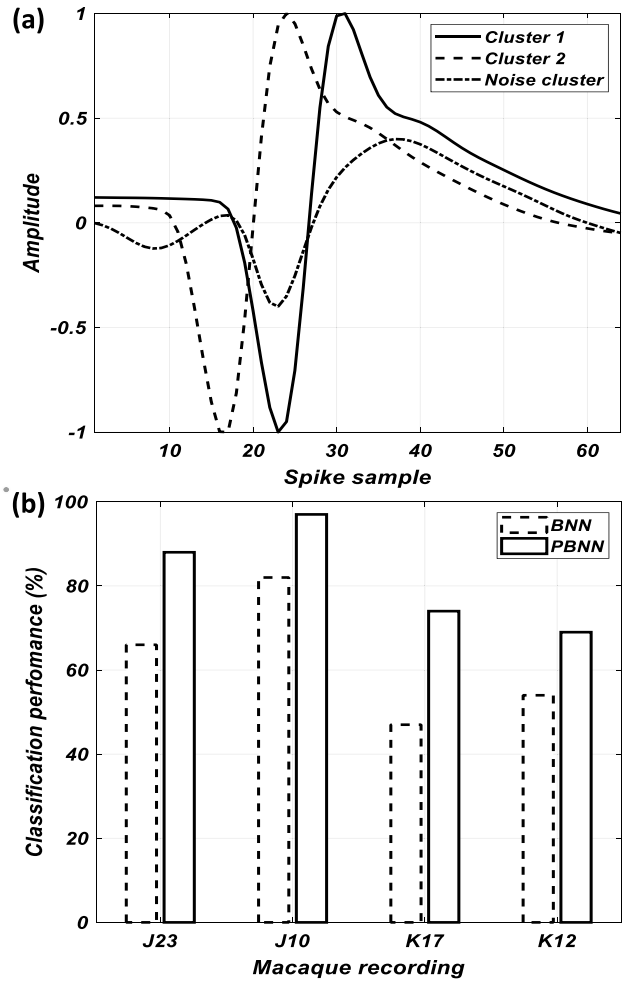

Fig. 4 **a** Example cluster waveforms of the J23 macaque recording and **b** the classification performance of the BNN and PBNN-based clustering over four macaque recordings

employed. The performance of the BNN and PBNN was assessed using the 20% testing subset. Figure 4(a) shows an example of the mean *k*-means cluster waveforms and (b) shows the performance that the classification performance of the PBNN outperforms that of the BNN over four macaque recordings.

4 Hardware architecture and implementation of the PBNN-based classifier

In our designed and implemented PBNN-based spike sorting system, the spike waveforms are first detected using the nonlinear energy operator (NEO) algorithm [34] and aligned to the maximum amplitude, as described in our earlier work in [7] and [11]. In our design, we implement the NEO unit using log-based approximate multipliers to conserve circuit area and power [35], which we refer to approximated NEO. To estimate a spike detection threshold, the noise of the approximated NEO signal is computed using the root-mean square (RMS) method [36]. The spike detection threshold is then set to a scaled value of the estimated noise, given as $4\sigma_e$, where σ_e denotes the RMS of the estimated noise. The hardware implementation details are given in [37]. Once a detected spike is aligned, the waveform is sent serially to the FE module designed based on the DD-2Ex algorithm, as shown in Fig. 5.

The FE module consists of a 7-word shift register for the scaling factors $\delta = 3$ and $\delta = 7$, and four sets of comparators and registers for finding the minimum and the maximum samples of $DD_{\delta=3}[n]$ and $DD_{\delta=7}[n]$. The registers R are updated only when the current value of the spike waveform is larger or smaller than the value currently stored in R for the maximum and the minimum features, respectively. After 64 clock cycles, the FE module will have computed the maximum and the minimum values of the $DD_{\delta=3}$ and $DD_{\delta=7}$ waveforms. These values are concatenated and passed on to the PBNN module via the FVO feature vector output port.

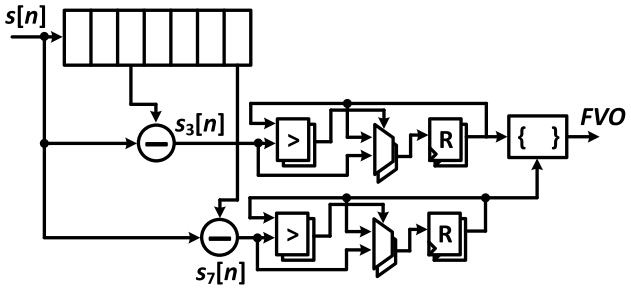

Fig. 5 The block diagram of the DD-2Ex-based FE module

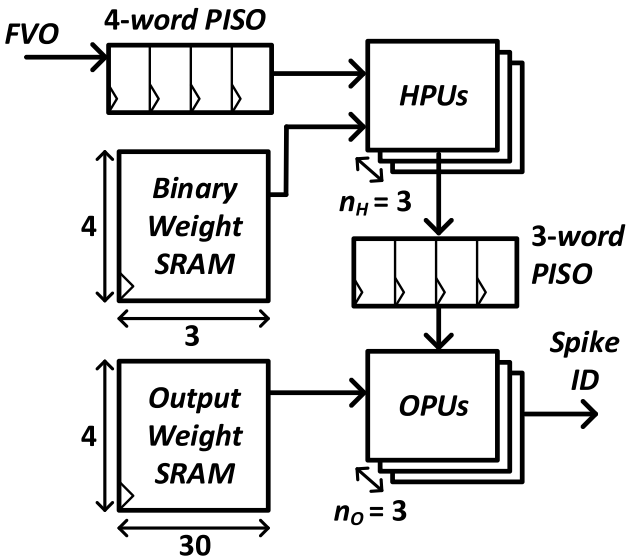

Fig. 6 The block diagram of the PBNN

The top-level block diagram of the designed PBNN is shown in Fig. 6. The datapath classifies feature vectors into their corresponding spike IDs. It consists of parallel-input serial-output PISO shift registers, hidden-layer processing units HPUs, the output layer processing units OPUs, and the SRAM-based memory units. The 4-word feature vector outputs FVOs from the FE module are passed on to the PBNN datapath, which are then stored in the 4-word PISO shift registers. Each word in the PISO is passed on to the HPUs serially to compute the hidden layer outputs. The Binary Weight SRAM stores the binarized weights of the hidden layer and consists of four words of three bits, one word per input feature. The output of the HPUs is stored in the 3-word PISO for serial processing by the OPUs. Because the binarization kernel is not applied to the output layer weights, they are represented in a 10-bit signed fixed-point format with 4 and 6 bits for the integer and the fractional parts, respectively.

The HPUs and OPUs shown in Fig. 7(a) and (b), respectively, are similar modules which compute the accumulated input activity of the previous layer. The HPUs accept the binarized weight values, which are used as the select lines of the multiplexers to choose either the sample of the feature vectors or its negated value in two's complement format. Because the binarization kernel is not applied to the output of the hidden layer, the HPUs accumulate the weighted input feature vectors using 19-bit accumulators. Each OPU accepts 10-bit signed weights and computes the product of the weights and the HPUs' outputs. To represent the accumulated input activity of the HPUs with sufficient resolution, the OPUs use 32-bit accumulators. The accumulated value is then passed on to the Qsigmoid activation function and is compared against the bias values for each AN stored in the Output Weight SRAM shown in Fig. 6. The Qsigmoid function will then generate spike IDs.

We have implemented the PBNN-based spike sorting system in a standard 180-nm CMOS process. The chip layout, shown in Fig. 8, is estimated to consume 2.5 μW of power from a 1.8-V supply while operating at 24 kHz and to occupy 0.34 mm^2 of silicon area. The NEO and RMS-based spike detection unit occupies 0.16 mm^2 of silicon area and consists of the approximated NEO unit, the adaptive RMS noise estimation unit, and the spike waveform alignment unit. The DD-2Ex feature extraction unit occupies 0.02 mm^2 of silicon area. The PBNN classifier occupies 0.15 mm^2 and consists of the Qsigmoid activation function for the output layer, an SRAM unit SRAMFP, which stores the output layer's weights in the fixed-point format, and the SRAM unit SRAMB, which stores the binarized weights for the input layer. The spike sorting system was described using Verilog HDL and the synthesis was performed using Synopsys DC Compiler. After synthesis, the placement and routing was done with Cadence Innovus. For estimating the power consumption, the post place and route synthesized netlist was used and the switching activity of the ASIC was modeled using the WaveClus datasets. Table 2 gives the power consumption of the various ASIC modules, assuming a mean spike firing rate of 10 Hz. The frequency column denotes the

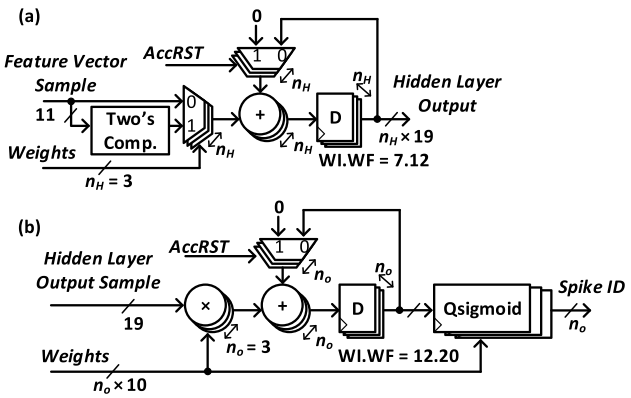

Fig. 7 The block diagram of **a** the hidden layer processing units (HPUs) and **b** the output layer processing units (OPUs)

Table 2 The power consumption of each ASIC module

Module	Power (μW)	Frequency (Hz)	Energy (nJ)
Spike detection	0.68	24000	672
Spike alignment	0.26	~10	8.77
Feature extraction	0.02		0.52
PBNN classifier	0.57		2.37

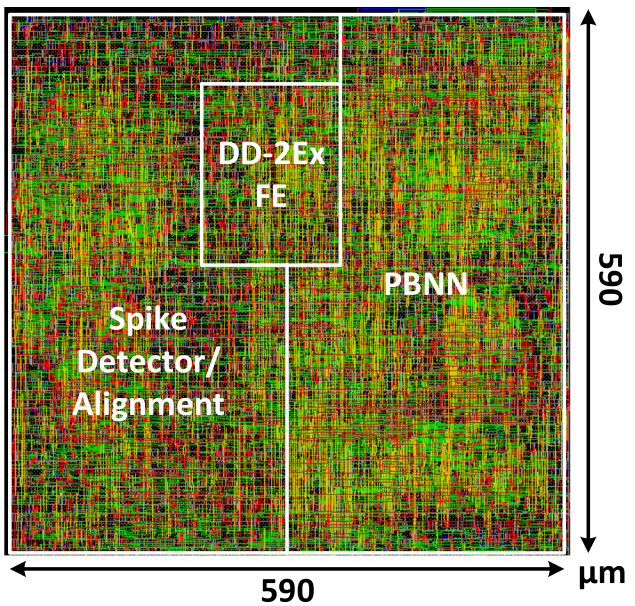

Fig. 8 The layout of the designed PBNN-based spike sorting system

rate at which the power is dissipated, i.e., for the spike detection it is dissipated on each input sample and for the subsequent units the power is dissipated on every spike event. It can be seen that the spike detection and noise estimation circuitries dissipate the most power and energy. It is also clear that the DD-2Ex feature extraction provides a very power and energy efficient operation. The energy shown in Table 2 is over one second of operation, and the total energy consumption is approximately 683.66 nJ.

Based on the results given in Table 1, although the performance of the *k*-medoids clustering is similar to that of the PBNN in Mode 00, to compare their hardware efficiency, we consider their computational complexities and their memory requirements based on three design parameters: the wordlength of the feature vectors w_k, the wordlength of the PBNN output layer parameters w_o, and the number of clusters N. The computational complexity of the PBNN and *k*-medoids can be compared based on the number of their operations. Given N clusters and four samples per feature vector, computing the Euclidean distance metric $\eta = \sqrt{\sum_{i=1}^{4}\left[x_i - y_i\right]^2}$ in the *k*-medoids algorithm, where x and y denote the template feature vector and the new feature vector, respectively, requires $7N$ additions and $4N$ multiplications, not considering the square root, which is unnecessary for comparisons. Comparing N values to one another would also require $N(N-1)/2$ comparisons, not accounting for the priority logic required to assign the feature vectors to only one cluster. Assuming that the complexity of an addition and a multiplication can be estimated as 2 and 4 times of a comparison, respectively [38], the computational complexity of the *k*-medoids algorithm is $30N + N(N-1)/2$ operations. For the PBNN, the multiplication with -1 at the input layer is realized using a two's complement operation, which requires 21 additions. The output layer's multiplication of a (1×3) vector and a $(3 \times N)$ weight matrix requires $3N$ multiplications and $2N$ additions. The QSigmoid function also requires N comparisons. The normalized computational complexity of the PBNN is thus $17N + 42$, which makes the PBNN algorithm computationally more efficient $N > 3$ clusters. The parameter memory stores the cluster centroids and weight matrices for the *k*-medoids and the PBNN algorithms, respectively. For the *k*-medoids algorithm, N template feature vectors are stored for comparison with the newly detected spike feature vectors, requiring a total of $4Nw_k$ bits. For the PBNN of size $4 \times 3 \times N$, the binarized input layer requires 12 bits and the output layer in Mode 00 requires $(N+1)w_o$ bits, a total of $(N+1)w_o + 12$ bits. Thus, the *k*-medoids algorithm will be more memory-efficient only if $w_k < (w_o(N+1) + 12)/4N$. Given that the analog-to-digital converters in BMIs are typically 10 to 16 bits and w_o ranges between 4 and 10 bits, the PBNN is also more memory-efficient than *k*-medoids.

Table 3 gives the ASIC characteristics and implementation results of various state-of-the-art spike sorting systems. In [8] we implemented an ANN-based spike sorting system composed of one hidden layer neuron and three output layer neurons utilizing the ReLU activation function. In [9] we implemented a BNN-based spike sorting system to classify spike waveforms in a standard 180-nm CMOS process. In [11] we designed and implemented an OSort-based spike sorting system in a standard 32-nm CMOS process. In [7] we designed and implemented a TM-based spike sorting ASIC in a 45-nm CMOS process. The work in [6] performs the NEO-based 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 sorting. The work in [15] uses the absolute value detection scheme and implements the OSort clustering algorithm for 16 channels. In [12], spikes

Table 3 The ASIC characteristics and implementation results of various spike sorting systems

Design	Ours	[8]	[9]	[11]	[7]	[6]	[15]	[12]	[13]	[10]	[14]	[39]	[40]
Algorithm	PBNN	ANN	BNN	OSort	TM	FE	OSort	OSort	FE	FE	TM	FE	FE
Median accuracy	0.91	0.98	0.91	0.87	0.90	0.77	0.75	0.93	0.84	0.85	0.93	0.86	0.92
Data rate reduction	2000×	1866×	2000×	1600×	3200×	11×	240×	278×	240×	–	–	257×	–
Adaptive spike detection	Y	N	N	N	N	N	Y	N	Y	Y	N	N	N
Supervised	Y	Y	Y	N	Y	Y	N	N	N	Y	N	N	N
Technology (nm)	180	32	180	32	45	90	65	45	45	130	40	65	65
Core voltage (V)	1.8	0.7	1.8	1.16	0.25	0.55	0.27	–	1.1	1.2	1.1	0.54	1
Frequency (kHz)	24	20	24	24	24	4000	480	56	960	160	500	3200	30
Norm. area/ch (mm^2)†	0.34	0.36	0.33	102.8	5.7	0.396	0.84	1.33	51.3	0.066	0.473	0.036	1.08
Norm. power/ch(μW)†	2.5	7.81	2.02	4.57	1.35	27.67	152.4	–	41.33	0.96	42.92	2.49	0.54
Norm. energy/ch (pJ)†‡	104.16	468.6	84.16	190.41	56.25	1152.9	6350	–	1722	40	1788	103.7	22.5

†Scaled to a 180-nm process with a 1.8 V supply voltage, as described in [41]

††Normalized to a clock frequency of 24 kHz

are first detected using a voltage threshold and aligned to maximum absolute amplitude clustered using OSort. The work in [13] performs spike sorting using 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 [10] presents a multi-channel spike sorting ASIC based on the Haar wavelet FE algorithm. The design in [14] presents a multi-channel TM-based spike sorting ASIC with a built-in OSort learning system. The work in [39] presents a multi-channel spike sorting processor based on the integer coefficient FE and clustering. Since the ASIC designs in [6, 10, 14, 15, 39] are multi-channel systems, for a fair comparison, their equivalent area and power consumption results for the single-channel sorting are given in Table 3. Finally, the work in [40] presents the ASIC implementation of a dictionary learning-based FE unit, in which the dictionary values are constrained to -1, 0, or 1 and no multiplications are used. However, the ASIC implementation is for the FE module only. We have also reported the normalized power consumption and area of the designs listed in Table 3 to 180-nm technology with a 1.8 V supply voltage, following the scheme presented in [41]. The spike sorting systems, which employ unsupervised learning, support real-time classification of detected spike waveforms. However, the TM-based and the neural network-based spike sorting systems that require pre-processing of the neural recordings in order to generate the template waveforms and synaptic weights, respectively, need a significantly smaller storage with a lower computational complexity, while providing comparable classification accuracy. Compared to the TM-based sorting, in which a distance metric is used to quantify the similarity between two spike waveforms, the neural network-based schemes can offer a more robust classification due to the non-linearity of the network model. Note that the work in [10] presents the most compact design, however, its median classification accuracy is relatively low at about 70% for low SNR neural data, while that of the PBNN-based classifier is 83%. It is shown that the designed PBNN-based spike sorting system not only offers similar accuracy to those of the state-of-the-art systems, also as opposed to the other designs for which performance degrades with increasing noise levels, it provides a robust classification accuracy over various noise levels and datasets.

Among the neural network-based architectures, one can see the advantage of reducing the computational complexity of the PBNN. Compared to our ANN classifier in [8], our BNN classifier in [9] reduced the area slightly, from $0.36mm^2$ to $0.33mm^2$, but reduces the power consumption by over two-fold. By employing a PBNN classifier, the size of the network and the number of parameters can be reduced. At the cost of slightly increased area (only $0.01mm^2$ larger than our BNN ASIC in [9]), the power of the classifier is reduced by a factor of 3.42. Assuming a 24 kHz sample rate with 10 bits per sample, the input rate 240 kbps. With an average neuron spiking rate of 40 spikes per second [42], representing the PBNN's classified outputs with three bits reduces the data rate to 120 bps. This results in a 99.9% reduction in the output data rate compared to the input sampling rate. Since the energy required to transmit one bit of data is approximately 3 nJ [43], the power consumption for the wireless transmission of the spike IDs is about 360nW. The total power dissipation of our synthesized ASIC design is thus 2.8μW with the power density of $8.23\mu W/mm^2$, which satisfies the tissue-safe requirement for the brain implantable devices [44].

5 Conclusion

This article presented the efficient hardware design and implementation of a spike sorting system utilizing a partially binarized neural network (PBNN) classifier. Among various efficient feature extraction algorithms for hardware realization, the discrete derivatives-based feature extraction algorithm was chosen and employed to reduce the dimensionality of neural spike waveforms and decrease the memory requirement of the neural network. It was shown that the designed spike sorting system not only offers similar accuracy to those of the state-of-the-art systems, also as opposed to the other designs for which performance degrades with increasing noise levels, it provides a robust classification accuracy over various noise levels. The implemented PBNN-based spike sorting system meets the strict power dissipation constraints of implantable devices, making it applicable in brain-machine interface systems.

Acknowledgements This work was supported by the Center for Neurotechnology (CNT), a National Science Foundation (NSF) Engineering Research Center (EEC-1028725), and by the NSF Award #2007131. The authors would like to thank Mitchell Timken for his assistance, and Dr. Eberhard Fetz and Dr. Steve Perlmutter at the Washington National Primate Research Center at the University of Washington for providing access to the macaque recordings.

Declarations

Conflict of interest The authors have no competing interests to declare.

Ethical approval This work does not contain any studies involving human participants or animals performed by the authors.

References

1. Willett FR, Avansino DT, Hochberg LR, Henderson JM, Shenoy KV. High-performance brain-to-text communication via handwriting. Nature. 2021;593(7858):249–54.

2. Chang EF, Anumanchipalli GK. Toward a speech neuroprosthesis. J Am Med Assoc. 2020;323(5):413–4.
3. Glaser JI, Benjamin AS, Chowdhury RH, Perich MG, Miller LE, Kording KP. "Machine learning for neural decoding," Eneuro, 2020; vol. 7, no. 4.
4. Lewicki MS. A review of methods of spike sorting: the detection and classification of neural action potentials. Network Comput Neural Syst. 1998;9(4):R53-78.
5. Kocaturk M, Gulcur HO, Canbeyli R. Toward building hybrid biological/in silico neural networks for motor neuroprosthetic control. Front Neurorobotics. 2015;9:8.
6. Karkare V, Gibson S, Markovic D. A 130-μw, 64-channel neural spike-sorting DSP chip. IEEE J Solid-State Circuits. 2011;46(5):1214–22.
7. Valencia D, Alimohammad A. An efficient hardware architecture for template matching-based spike sorting. IEEE Trans Biomed Circuits Syst. 2019;13(3):481–92.
8. Valencia D, Thies J, Alimohammad A. Frameworks for efficient brain-computer interfacing. IEEE Trans Biomed Circuits Syst. 2019;13(6):1714–22.
9. Valencia D, Alimohammad A. Neural spike sorting using binarized neural networks. IEEE Trans Neural Syst Rehabil Eng. 2021;29:206–14.
10. Yang Y, Boling S, Mason A. A hardware-efficient scalable spike sorting neural signal processor module for implantable high-channel-count brain machine interfaces. IEEE Trans Biomed Circuits Syst. 2017;11(4):743–54.
11. Valencia D, Alimohammad A. A real-time spike sorting system using parallel osort clustering. IEEE Trans Biomed Circuits Syst. 2019;13(6):1700–13.
12. Liu Y, Sheng J, Herbordt MC. "A hardware design for in-brain neural spike sorting," in IEEE High Performance Extreme Computing Conference, 2016;1–6.
13. Zamani M, Jiang D, Demosthenous A. An adaptive neural spike processor with embedded active learning for improved unsupervised sorting accuracy. IEEE Trans Biomed Circuits Syst. 2018;12(3):665–76.
14. Xu H, et al. Unsupervised and real-time spike sorting chip for neural signal processing in hippocampal prosthesis. J Neurosci Methods. 2019;311:111–21.
15. Karkare V, Gibson S, Markovic D. A 75-μw, 16-channel neural spike-sorting processor with unsupervised clustering. IEEE J Solid-State Circuits. 2013;48(9):2230–8.
16. Hyvärinen A, Oja E. Independent component analysis: algorithms and applications. Neural Netw. 2000;13(4–5):411–30.
17. Korat UA, Alimohammad A. A reconfigurable hardware architecture for principal component analysis. Circuits Syst Signal Process. 2019;38(5):2097–113.
18. Lopes MV, Aguiar E, Santana E, Santana E, Barros AK. "ICA feature extraction for spike sorting of single-channel records," in IEEE biosignals and biorobotics conference, 2013;1–5.
19. Awais MK, Andrew JM. "On-chip feature extraction for spike sorting in high density implantable neural recording systems," in IEEE biomedical circuits and systems conference, 2010;13–6.
20. Zviagintsev A, Perelman Y, Ginosar R. "Low-power architectures for spike sorting," in IEEE EMBS conference on neural engineering, 2005;162–5.
21. Li P, Liu M, Zhang X, Chen H. "Efficient online feature extraction algorithm for spike sorting in a multichannel FPGA-based neural recording system," in IEEE biomedical circuits and systems conference, 2014;1–4.
22. Nadasdy Z, Quiroga RQ, Ben-Shaul Y, Pesaran B, Wagenaar DA, Andersen RA. "Comparison of unsupervised algorithms for on-line and off-line spike sorting," in Proceedings of the annual meeting of the society for neuroscience, 2002.
23. Gibson S, Judy JW, Markovic D. Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. IEEE Trans Neural Syst Rehabil Eng. 2010;18(5):469–78.
24. Paraskevopoulou SE, Barsakcioglu DY, Saberi MR, Eftekhar A, Constandinou TG. Feature extraction using first and second derivative extrema for real-time and hardware-efficient spike sorting. J Neurosci Methods. 2013;215(1):29–37.
25. Zamani M, Demosthenous A. Feature extraction using extrema sampling of discrete derivatives for spike sorting in implantable upper-limb neural prostheses. IEEE Trans Neural Syst Rehabil Eng. 2014;22(4):716–26.
26. Lilliefors HW. On the Kolmogorov–Smirnov test for normality with mean and variance unknown. J Am Stat Assoc. 1967;62(318):399–402.
27. Valencia D, Fard SF, Alimohammad A. "An artificial neural network processor with a custom instruction set architecture for embedded applications," IEEE Transactions on circuits and systems I: regular papers, 2020;1–11.
28. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. "Binarized neural networks," in Advances in neural information processing systems, 2016;4107–15.
29. Simons T, Lee D-J. A review of binarized neural networks. Electronics. 2019;8(6):661.
30. Quiroga R, Nadasdy Z, Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Comput. 2004;16(8):1661–87.
31. Geiger L, Team P. Larq: an open-source library for training binarized neural networks. J Open Sour Softw. 2020;5(45):1746. https://doi.org/10.21105/joss.01746.
32. W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in Proceedings of the AAAI conference on artificial intelligence, vol. 31, no. 1, 2017.
33. Pedreira C, Martinez J, Ison MJ, Quiroga RQ. How many neurons can we see with current spike sorting algorithms? J Neurosci Methods. 2012;211(1):58–65.
34. Kaiser JF. "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;381–4.
35. Kim MS, Del Barrio AA, Oliveira LT, Hermida R, Bagherzadeh N. Efficient mitchell's approximate log multipliers for convolutional neural networks. IEEE Trans Comput. 2018;68(5):660–75.
36. Guillory K, Normann R. A 100-channel system for real time detection and storage of extracellular spike waveforms. J Neurosci Methods. 1999;91(1–2):21–9.
37. Valencia D, Mercier PP, Alimohammad A. In vivo neural spike detection with adaptive noise estimation. J Neural Eng. 2022;19(4): 046018.
38. Mora-Mora H, Mora-Pascual J, García-Chamizo JM, Jimeno-Morenilla A. Real-time arithmetic unit. Real-Time Syst. 2006;34(1):53–79.
39. Do A, 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 Trans VLSI Syst. 2019;27(1):126–37.
40. Zamani M, Sokolić J, Jiang D, Renna F, Rodrigues MR, Demosthenous A. Accurate, very low computational complexity spike sorting using unsupervised matched subspace learning. IEEE Trans Biomed Circuits Syst. 2020;14(2):221–31.
41. Stillmaker A, Xiao Z, Baas B, "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, 2011;4,m8.
42. Gibson S. "Neural spike sorting in hardware: From theory to practice," Ph.D. dissertation, University of California Los Angeles, 2012.

43. Chen F, Chandrakasan AP, Stojanovic VM. Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors. IEEE J Solid-State Circuits. 2012;47(3):744–56.
44. Kim S, Tathireddy P, Normann RA, Solzbacher F. Thermal impact of an active 3-D microelectrode array implanted in the brain. IEEE Trans Neural Syst Rehabil Eng. 2007;15(4):493–501.