Chapter 2
Energy-Efficient Digital Processing for Neural Action Potentials

Vaibhav Karkare, Sarah Gibson, and Dejan Marković

Abstract This chapter discusses algorithm, architecture, and circuit techniques for efficient implementation of neural signal processing circuits. In particular, the focus is on spike sorting and compressive sampling for action potentials. The chapter begins with an introduction to spike sorting and compressive sampling, and the need for their implementation in modern-day neural recording systems. We then illustrate, through examples, some useful methods for algorithm selection and optimization. Digital design techniques that are beneficial in power and area reduction for neural signal processing DSPs are also discussed. Finally, we discuss the challenges and future directions in the area of biosignal processing.

2.1 Introduction

Spike sorting and compressive sampling (CS) are popular processing techniques to provide reduction in the output data rates for neural recording systems. Spike sorting, being a lossy compression technique, is suitable for real-time applications like brain–machine interfaces (BMI). On the other hand, CS is suited for neuroscientific applications that require access to raw recorded data at all times. In order to address data-rate reduction requirements (the need for data-rate reduction would be explained soon) for both these applications, we discuss the implementation of spike sorting and compressed sensing for neural recording systems.

V. Karkare • S. Gibson • D. Marković (✉)
Department of Electrical Engineering, University of California, Los Angeles, CA, USA
e-mail: dejan@ee.ucla.edu

Z. Yang (ed.), *Neural Computation, Neural Devices, and Neural Prosthesis*,
DOI 10.1007/978-1-4614-8151-5_2,

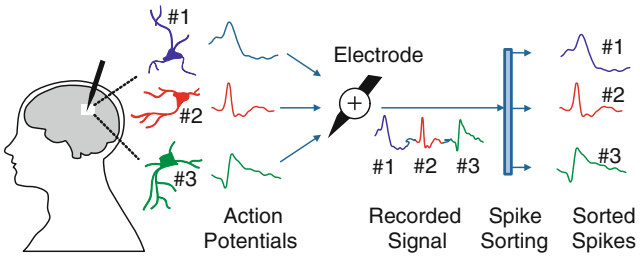

Fig. 2.1 Each electrode in an implanted electrode array records signals from multiple neurons. For many applications, it is required to classify these recorded signals according to their source neurons. This process of classifying the recorded action potentials according to their source neurons is called spike sorting

2.1.1 Spike Sorting

It is well known that neurons communicate with each other using electrical signals known as "action potentials" or "spikes." In the recent past, significant progress has been made in our understanding of the brain, owed largely to the analysis of electrical signals recorded from the brain. In modern neuroscientific studies and clinical procedures, neural signals are recorded from the brain using implanted electrode arrays. Modern electrode arrays [1] consist of hundreds of electrodes. Each electrode in these electrode arrays records signals from multiple neurons (Fig. 2.1).

While analyzing the collective signal of a group of neurons (multi-unit recording) is interesting for some studies, many neuroscientific analyses require knowledge of single-neuron, or single-unit, activity. For instance, correlations between an applied stimulus and the activity of an individual neuron or the correlation between the activity of different individual neurons can only be observed from single-unit neural recordings. Further, single-unit neural recordings are shown to significantly enhance the performance of brain–machine interfaces [2]. Therefore, it is important to classify the multi-unit action potentials recorded by a single electrode based on their source neurons. This process of classifying the recorded action potentials according to the neurons from which they originate is called *spike sorting*. Before we delve into the implementation details of spike sorting hardware, we would first present a brief review of the steps involved in spike sorting and the need for designing energy-efficient spike sorting hardware.

Recorded neural data contains action potentials and local field potentials (commonly referred to as LFPs). LFPs have a higher amplitude (of around 2–6 mV) compared to that of action potentials, which have amplitudes ranging from 10 μV to 1 mV. LFPs occupy a frequency band of 0.5–250 Hz, while action potentials typically occupy a range of 300 Hz to 3 kHz. Raw neural data recorded from the

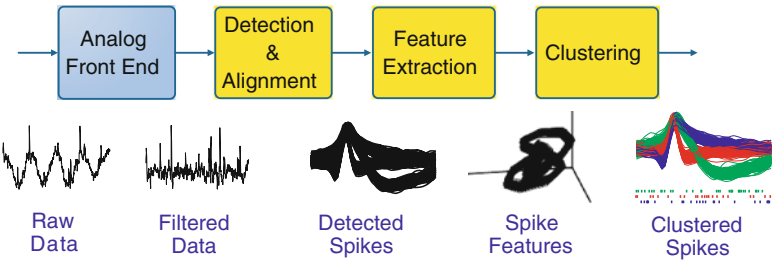

Fig. 2.2 Spike-sorting process

brain is amplified, digitized, and high-pass filtered to remove the LFP.[1] These operations are grouped under the "analog front end" module in Fig. 2.2. Spike sorting is performed on this filtered and digitized neural data.

Spike sorting can be divided into three major steps (Fig. 2.2): (1) detection and alignment, (2) feature extraction (FE), and (3) clustering.

The first spike-sorting step is spike detection. As the name indicates, the objective of this step is to search for action potentials in the raw neural data. This is typically done by placing a threshold on the raw data or on a signal derived from the raw data. Several detection methods are used in the neuroscience community which differ from each other in the signal transformation applied and the thresholding method used. After detection, the recorded spikes are aligned to a common reference such as the maximum spike amplitude or maximum spike derivative in order to avoid mis-aligned spikes from the same neuron being classified into different groups. The second major spike-sorting step is called feature extraction (FE). FE is the process of transforming the recorded action potentials into a domain that better separates them from each other, making classification easier. FE also reduces the number of samples that needs to be processed per spike, thus reducing the computational complexity of the processing that follows. Principal component analysis [3], discrete derivatives [4], discrete wavelet transform [5], and the integral transform [6] are some examples of the transformations used for feature extraction. The extracted features of the recorded action potentials are finally passed through the clustering step, where the input spike features are grouped into clusters. Most clustering methods require storage of all the data to be classified, in order to map each spike to its class. The popular K-means and fuzzy c-means are examples of such clustering methods. Most of these clustering methods also require a user input for the number of clusters present in the data. Since the precise number of neurons is not known, the user often has to guess the number of clusters present in the data, which makes the entire clustering process susceptible to human errors. In order to avoid these limitations, unsupervised clustering algorithms like super-paramagnetic clustering [7] and online clustering [8] can be used.

[1] Local field potentials (LFPs) are of interest to clinicians for several studies like diagnosis of epileptic patients. However, the focus of this chapter is spike sorting. Hence, we assume LFP to be filtered immediately after the signal is recorded.

The process of neural recording, followed by spike sorting, is currently performed using a rudimentary setup. In the traditional neural recording setup, the recorded signals are carried outside the body by transcutaneous wires to bulky amplifiers and ADCs, mounted on shelves. Signal processing, including spike sorting, is performed off-line, in software. This setup for neural recording has several disadvantages. The wires limit the freedom of movement of the subject and increase the risk of injury and infection. The signal quality is also degraded due to increased motion artifacts. Implantable, integrated wireless neural recording systems can solve these problems. The goal for the development of an implanted wireless recording system is to integrate all the system components on a chip that can be fabricated at the base of the implanted electrode array. Such implanted electronic devices have to meet stringent power density constraints. The power density needs to be much less than 800 $\mu W/mm^2$, the power density known to damage brain cells [9]. The area should also be minimized to allow for integration with the implanted electrode arrays. Although the system is subject to stringent power constraints, it also needs to transmit exorbitantly high data rates if raw data is transmitted for all the channels. For instance, a data rate of 11 Mbps is required to transmit raw data for a 64-channel system. Transmission of such high data rates is not feasible while meeting the power constraints imposed on implantable devices. Therefore, the data must be processed on-chip to reduce the output data rate.

The process of spike sorting, which is primarily a functional requirement has an added advantage: It can reduce the output data rate by more than 200 times.[2] By transmitting the sorting results instead of raw data, the output data rate for a 64-channel system, for example, can be reduced from 11 Mbps to a manageable 50 kbps. It is, therefore, desirable to perform on-chip spike sorting not only to meet the functional requirement of real-time processing but also to reduce the output data rates required for transmission. If transmission of spike IDs (the final output of spike sorting) is not acceptable, a data-rate reduction of about 10 times can still be obtained by transmitting only the detected spikes rather than all the recorded data. Inclusion of on-site or on-chip spike sorting thus reduces the output data rates and hence the system power consumption. This makes it feasible to support a high-channel-count wireless neural recording and telemetry system.

2.1.2 Compressive Sampling

As we discussed in the previous section, spike sorting can be used to reduce the data rate. Transmitting the output of a spike-sorting chip, the spike features or IDs, would potentially reduce the data rate by two orders of magnitude compared to

[2] The data-rate reduction numbers correspond to a 64-channel system with a sampling rate of 24 kSa/s. The typical action potential spans 48 samples and the spike firing rate is assumed to be 100 spikes/second.

transmitting raw data, but it would mean that the full action potential waveforms would not be available for analysis. This may not be acceptable in various studies that require the spike morphologies of the recorded action potentials in addition to the spike classification results. For instance, the individual spike waveforms can be used to characterize the type of neuron and possibly to distinguish between principal (primarily excitatory) and non-principal (largely inhibitory) cells for epilepsy studies. Other studies require spike widths (with varying definitions of widths), ratios of ascending to descending slopes of the spikes [10], etc. Neuroscientists often revisit previously recorded data to test new hypotheses, and thus may require different features to be extracted from the recorded spikes. Therefore, there is a need to reduce the transmitted data rate while allowing access to the recorded neural action potentials.

Compressive sensing (CS) is a recently developed theory that enables signal reconstruction from a small number of non-adaptively acquired sample measurements corresponding to the information content of the signal rather than to its bandwidth [11]. Information content or sparsity is quantified by estimating the number of the significant coefficients when the signal is projected into a space that accentuates its principal components. Therefore, if action potentials are sparse, compressive sensing would allow us to reduce communication costs and bandwidth compared to transmitting raw action potentials acquired at the Nyquist rate.

For the specific case of action potentials, we can modify the conventional data recovery procedure in CS to be able to obtain higher reductions in output data rate. We will describe these modifications in the following section. Before we proceed to the modifications, we will first provide a brief introduction to compressive sampling.

Let the signal corresponding to an aligned spike waveform be $x \in \mathbb{R}^n$, where n is the length of the window within which the spike is completely contained. We apply "soft compressive sensing"[3] to this signal window to generate a set of m measurements for each spike $y = \Phi x$, where $m < n$ and $\Phi \in \mathbb{R}^{m \times n}$ is a sampling matrix that performs an arbitrary linear projection. The objective of our wireless neural recording system is to recover the stream of spikes from these compressed measurements transmitted over a low-power radio. For this recovery, we require the spike waveforms to be sparse or compressible in some domain. We found [12] that the discrete wavelet transform (DWT) with Daubechies filters to be suitable for compressing neural action potentials.

The spike is compressible in the DWT domain such that $z = \Psi x$ has few significant coefficients, where $\Psi \in \mathbb{R}^{n \times n}$ corresponds to the DWT operation. Significant coefficients are those that account for most of the signal energy.

[3] We refer to the process of applying random linear projections to a signal after sampling at the Nyquist rate as soft(-ware) CS to distinguish it from hard(-ware) CS, where the projections or their equivalent are performed in the analog or physical domain before sampling or digitization.

In particular, we locate the smallest set of DWT values that retains 99 % of the ℓ_2-norm of the spike. We term this set of DWT coefficients as the spike's "support." That is, we find the smallest set T such that:

$$\text{supp}(z) = T \; \|z_T\|_2 \geq C\|z\|_2 \tag{2.1}$$

where z_T is an approximation of z with only the terms in the set T and $C = 0.99$.

We can now formulate our recovery procedure as the basis pursuit denoising (BPDN) [13] problem:

$$\hat{z} = \underset{\tilde{z}}{\text{argmin}} \frac{1}{2} \left\| y - \Phi\Psi^{-1}\tilde{z} \right\|_2^2 + \lambda \|\tilde{z}\|_1 \tag{2.2}$$

where λ sets the significance of the sparsity with respect to the first noise tolerance term. It has been shown that when the ensemble matrix $\Phi\Psi^{-1}$ satisfies a condition known as the restricted isometry property (RIP) [14], the error in the solution to the above problem will be stable and bounded with overwhelming probability.

In [15], Lu and Vaswani introduced an approach to BPDN when additional knowledge is available. Specifically, they show that if the support of the spike waveform (or a part thereof) was known a priori, the error in the solution to Eq. (2.2) admits a lower bound. Their BPDN approach is given by:

$$\hat{z} = \underset{\tilde{z}}{\text{argmin}} \frac{1}{2} \left\| y - \Phi\Psi^{-1}\tilde{z} \right\|_2^2 + \lambda \|\tilde{z}_{T^c}\|_1 \tag{2.3}$$

where T^c is the complement of the known support and $\tilde{z}_{T^c}$ denotes the elements in $\tilde{z}$ that are not included within T. The bound on the ℓ_2 norm of the solution error depends on not only λ and T but also Δ—the part of the support that is unknown and Δ_e—the part of known support that is incorrect. If the true support of the signal can be denoted by N, the relationship between these sets of supports is $N = T \cup \Delta \backslash \Delta_e$. Lu [15] demonstrated that as the size of Δ reduces, the solution error decreases dramatically, especially at high compression ratios (i.e., $n/m \gg 1$). An intuitive way of looking at modified BPDN is that it searches for a solution that sparsifies the nonsignificant coefficients of the signal since these would have lower energy than all the coefficients considered together.

Having introduced spike sorting and compressive sampling, let us now discuss the design techniques that would allow us to build efficient spike-sorting DSPs.

2.2 Algorithm Selection and Optimization

In this section, we will first discuss the algorithm-level optimizations for spike sorting followed by optimizations for compressed sensing.

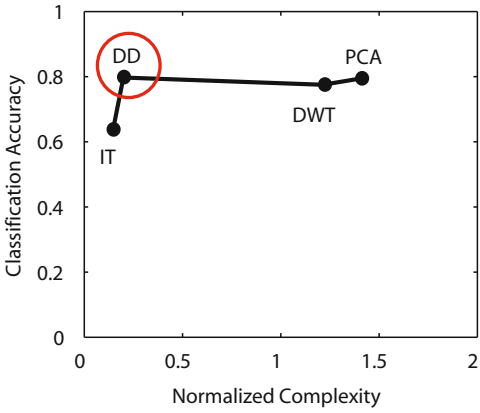

Fig. 2.3 Complexity–accuracy tradeoffs for feature-extraction algorithms

2.2.1 Spike-Sorting Algorithms

The first step to an efficient DSP implementation is to select the algorithms that are most suited for hardware implementation, while meeting the required performance criterion. As mentioned earlier, there are several algorithms in literature [16] for the various spike-sorting steps illustrated in Fig. 2.2. In [17], we evaluated the complexity–performance tradeoffs for these algorithms using the probability of detection, probability of false alarm, and classification accuracy as metrics for algorithm performance. The complexity of each algorithm was evaluated in terms of the number of equivalent additions required for the algorithm and the estimated memory requirement. Since the "ground truth" for actual recordings is never known, such an analysis has to be validated using simulated neural data sets across a wide range of spike shapes and SNRs. As an illustration of this analysis, Fig. 2.3 shows the median classification accuracy (for over 1,600 data sets of simulated neural data over an SNR range of −15–20 dB) versus computational complexity for four different feature-extraction algorithms: principal component analysis (PCA) [3], discrete derivatives (DD) [4], discrete wavelet transform (DWT), [5] and integral transform (IT) [6].

The normalized computational complexity is defined to be the sum of area normalized to the maximum area and operations per second (OPS) normalized to the maximum OPS among the algorithms considered. An operation was defined to be an eight-bit addition for this analysis. The expression for the normalized complexity is as follows:

$$\text{Normalized Complexity} = \frac{\text{OPS}}{\max(\text{OPS})} + \frac{\text{Area}}{\max(\text{Area})}. \tag{2.4}$$

Once a plot for complexity-performance tradeoffs is generated, the algorithm at the knee point of the curve can be identified as the complexity–accuracy-optimal algorithm, for example. Thus, from Fig. 2.3, we would choose the discrete-derivatives algorithm as our feature-extraction method. Based on a similar analysis,

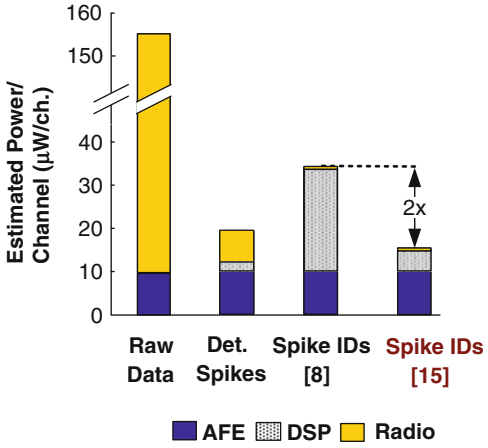

Fig. 2.4 The total system power increases with the implementation of online clustering [8]. However, appropriate modifications to online clustering for a multi-channel implementation [20] allowed us to reduce the total system power by 2x. Following values are assumed for this plot: AFE and ADC: 10 µW/channel, Spike Detection: 2 µW/channel, Online Clustering Implementation: 23.5 µW/channel, Transmitter: 6 nJ/byte

we recommended [17] using the nonlinear energy operator (NEO) [18] for detection and maximum derivative for alignment [19].

While the above analysis works well for detection and feature extraction, in the domain of clustering algorithms there is a dearth of algorithms that can be implemented in real-time hardware. This is because most clustering algorithms [7, 16], designed primarily for software implementations, require storage of all the data to be clustered. An exception to this general trend is the online clustering algorithm [8], which uses an on-the-fly, iterative process to handle real-time data streams. This algorithm allows us to meet the functional requirement of implementing real-time clustering hardware. However, it does not meet the requirement of reducing the system power consumption to support a higher channel count [20]. As shown in Fig. 2.4, the total system power increases with an implementation of the online clustering algorithm when compared to transmission of the detected spikes. Although the reduction in data rates after clustering reduces the power consumption in the radio, the high power consumed in the DSP increases the total system power. In [20] we showed how the online clustering algorithm can be suitably modified for a multi-channel implementation to reduce the DSP power by 5 times, thus lowering the total system power by 2 times compared to implementing the original algorithm.

The online clustering algorithm [8] relies on the computation of Euclidean distance between an incoming spike and the existing cluster means. The computed Euclidean distance is compared to a threshold (that is derived from the noise variance of the data). If the distance between the incoming spike and the existing cluster mean is greater than

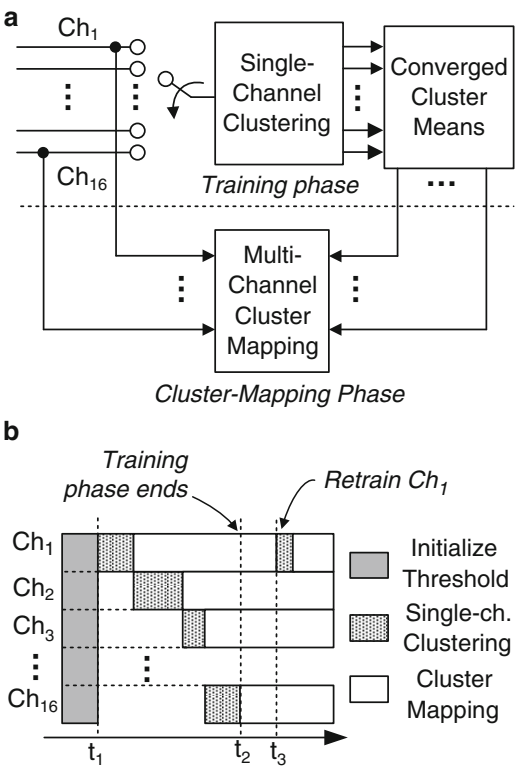

Fig. 2.5 (**a**) Two-phase implementation of the multi-channel online clustering algorithm. Cluster means are sequentially identified on each channel in the training phase and used in the cluster mapping phase to simultaneously classify the action potentials on all the channels. (**b**) Illustration of scheduling for the two-phase clustering algorithm

the threshold, the new spike forms a new cluster. However, if the distance is less than the threshold, the spike is assigned into an existing cluster and the mean is updated to be the weighted average of the incoming spike and the spikes already present in the cluster. The algorithm starts off with a large number (approximately 50) of clusters being formed which then get assimilated into, typically, less than six clusters. The power consumed by an implementation of the algorithm is dominated by the memory required to save the large number of transient clusters. In order to save the total memory in a multi-channel implementation, the algorithm can be split into two phases, as shown in Fig. 2.5. In the first, training phase, the channels are sequentially processed and the converged means on each channel are identified and saved in the memory. In the second, mapping phase, the incoming spikes are assigned to one of the cluster means that were identified in the first step. Splitting the clustering algorithm into two phases means that the large transient memory is required only for a single channel at a time. This reduces the total memory required by a factor of 6 in a 16-channel implementation.

Further simplifications to the algorithm can also be made based on its transient behavior. For example, during the training phase the cluster means are updated as the weighted average of the incoming spike and the existing cluster means. As more spikes get assigned to a given cluster, the cluster mean is not perturbed by a significant

amount and can be assumed to be a constant. Empirically, the cluster mean of a particular cluster can be considered to be a constant after about 30 spikes have been assigned to the cluster. The cluster mean convergence also determines the limit on the total number of spikes that need to be processed during the training phase.

2.2.2 *Compressed Sensing Algorithms*

In Sect. 2.1.2, we introduced compressed sensing as a method for reducing the output data rate for neural recording systems. However, in the particular case of action potentials the basis pursuit (BP) method of signal reconstruction can be modified in order to provide higher data-rate reduction. In conventional BP, the reconstruction method needs to find the support for the algorithm in a "blind" fashion. However, while recording neural action potentials, we know that the signals are being recorded from a finite population of neurons that remains relatively stable over time. This knowledge can be used to prefer solutions whose support (defined in Sect. 2.1.2) matches the support of pre-identified spikes. Thus, in [12] we proposed a recovery technique based on the learned union of supports that allowed us to reduce the output data rates by using the pre-identified supports.

Ideally, we would have liked to learn the support of each unique morphology discovered at an electrode and switch supports to the one being recovered. This would ensure that even if there are multiple models of the signal being recovered, the correct model would be used during reconstruction. While learning the different supports over time is quite feasible, knowing which spike support to use would require computationally expensive encoder involvement. Instead, we propose performing a set union over the learned supports and furnishing Eq. (2.3) with this set as T. The learning is continuous as the support of any newly recovered spikes is added to the union.

The procedure for union of supports recovery is outlined as follows: The decoder is initialized with an empty union set, $T^{(0)} = \emptyset$. When measurements for the first spike are received, the decoder uses Eq. (2.3) to recover it. With an empty T, this is equivalent to using conventional BPDN Eq. (2.2). After recovery, the support from this first reconstructed spike becomes the updated union set, $T^{(1)} = \text{supp}\left(\hat{z}^{(1)}\right)$. When measurements for the second spike are received, the decoder performs modified BPDN using Eq. (2.3) with the newly formed support set. Afterward, the support for this second reconstructed spike is computed and added to the union set, $T^{(2)} = T^{(1)} \cup \text{supp}(\hat{z}^{(2)})$. This process is repeated for subsequent spike measurements. Further details on the justification of using the learned union of supports can be found in [12].

In order to demonstrate the accuracy of CS reconstruction, we computed the median SNDR over more than 600,000 spikes from human electrophysiological recordings. The SNDR versus the number of CS measurements for different reconstruction methods is shown in Fig. 2.6. This plot shows that the signal recovered using the union of supports method has an SNDR that is on average

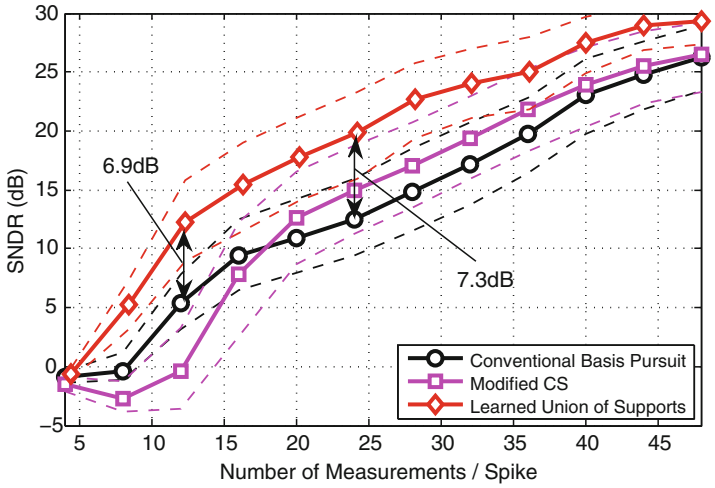

Fig. 2.6 Performance comparison of spike recovery using conventional basis pursuit, support from preceding spike, and learned union of supports. *Points* represent median SNDR and *dashed lines* represent the first and the third quartiles

5.4 dB greater (maximum 7.8 dB greater) than the SNDR of the signal recovered using the conventional basis pursuit recovery for the same number of CS measurements. We can also see that a signal reconstruction with 20-dB SNDR is possible with only 24 CS measurements per spike (where each spike originally had 48 samples). This implies that the data rate is 2 times lower than that required for transmission of detected action potentials.

Besides SNDR, it is also important to evaluate the performance of reconstruction in terms of classification accuracy (CA) of the acquired action potentials. Towards this purpose, we clustered the action potentials using the Osort spike-sorting software package. The clustering process was repeated for the signals reconstructed from a different number of CS measurements, ranging from 4 to 48, using each of the three reconstruction methods. The classification accuracy of the reconstructed spikes was computed by comparing the clustering results for each case with the clustering results of the original action potential waveforms. Figure 2.7 shows the median classification accuracy over the entire set of spikes analyzed for each of the three reconstruction methods. We find that the union of supports provides a higher classification accuracy than conventional basis pursuit and modified CS recovery. The classification accuracy for union of supports reconstruction reaches 80 % at 12 CS measurements, after which the classification accuracy increases very slowly with the number of measurements. This relationship between the CA and the number of measurements follows from the behavior of CS reconstruction, which is grossly inaccurate until it reaches a factor proportional to the signal sparsity. Beyond this point, the reconstruction is accurate and improves only slightly with an increased number of measurements. For 24 CS measurements, a median classification accuracy of more than 90 % is achieved.

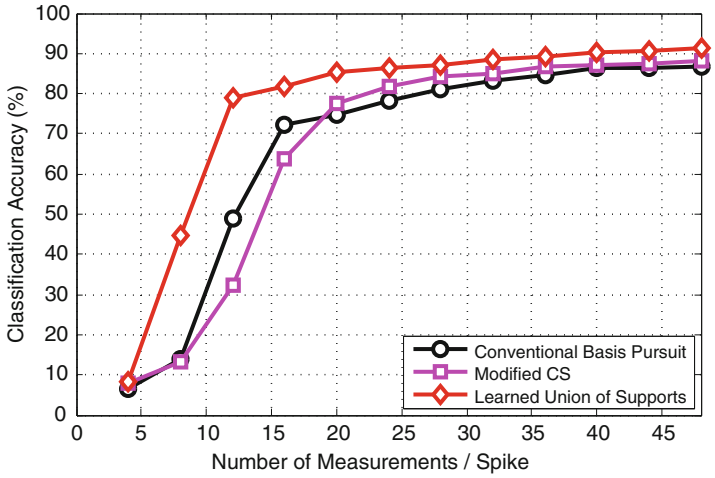

Fig. 2.7 Median classification accuracy versus number of CS measurements for basis pursuit, modified CS, and union of supports reconstruction

The purpose of the above example is to illustrate the kind of algorithm-level modifications that can be made to a spike-sorting algorithms and reconstruction techniques for CS to reduce the hardware cost. This example emphasizes the importance of being conscious of the final digital hardware at the algorithm design stage. In the following section, we will focus on the architecture- and circuit-level techniques that can be used to optimize the DSP.

2.3 Digital Design Techniques for Spike-Sorting DSPs

Spike-sorting DSPs receive input data at the rate of a tens of kHz per channel. The modern-day CMOS process, on the other hand, is capable of operating at GHz rates. Most spike-sorting DSPs are memory intensive, as opposed to the logic-intensive conventional DSPs. The slow data rates and register-dominated nature of spike-sorting DSPs introduce design tradeoffs that are not common to conventional signal processing chips.

Figure 2.8 shows the normalized energy per channel versus the normalized delay for the spike-sorting DSP core published in [22]. This DSP has a critical-path delay of 20 ns at the nominal supply voltage. This implies that the design at the minimum delay point (MDP) is 2,000× faster than the sampling-rate requirement. The energy–delay curve shown in Fig. 2.8 is plotted assuming that the DSP is operated at the maximum possible frequency at each voltage. Because the application delay is fixed, however, there is no reward for early computation, as the circuit continues to leak for the remainder of the clock cycle. Operating the DSP at the nominal supply voltage of 1.2 V puts the design at a high-energy point at which the DSP is heavily

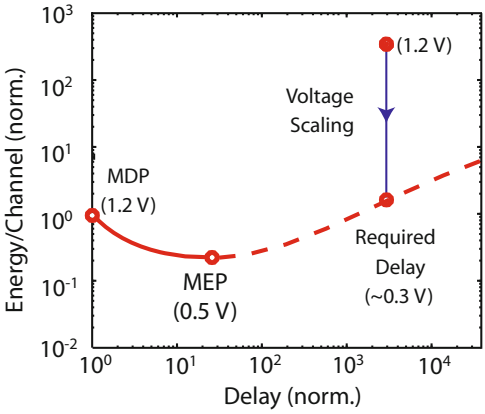

Fig. 2.8 Energy–Delay tradeoff of the spike-sorting DSP core that the authors published in [21]

leakage-dominated. In order to reduce the energy consumed, supply-voltage scaling can be used to bring the design from the high-energy point at 1.2 V to a much lower energy at 0.3 V. However, mere supply voltage scaling for a single-channel DSP places the design at a sub-optimal point, at which both the energy and the delay are greater than they are at the minimum-energy point (MEP) for the design. The sub-optimal region is indicated by a dotted line in Fig. 2.8. To bring the DSP to a desirable operating point between the minimum-delay and minimum-energy points, the designer can interleave the single-channel architecture. This method is, in fact, beneficial not only for spike sorting but also for most digital biosignal processing circuits. Since the input sample rates for these circuits are not high, it is advisable to scale the supply voltage aggressively and interleave multiple channels to minimize the penalty in leakage power.

Looking at the E-D tradeoff curve in Fig. 2.8, it might be tempting for a designer to interleave all the channels in the recording system into a single core. However, the maximum number of channels that one can interleave is limited due to the register-dominated nature of spike-sorting DSPs. Figure 2.9a shows the single-channel implementation of the NEO detection operation [18] and the accumulation of $\Psi(n)$ (the metric for energy in a spike $x(n)$). When this circuit is interleaved to support two channels as in Fig. 2.9b, interleaving registers are needed to ensure that the functionality is retained when an upsampled, interleaved stream of data is fed to the input. It can be seen from this figure that the total combinational logic hardware reduces by a factor of 2 in the interleaved implementation. However, the total number of registers in the design remains unchanged. In a fully parallel design, these registers only expend energy to shift data for a given channel. On the other hand, in an interleaved design, the same number of registers have to expend energy for data-shifting for all channels interleaved in a single core. In addition, interleaving also increases the switching activity, since it breaks the correlation between the data channels. Therefore, beyond 16-channel interleaving, the energy per channel actually starts to increase due to an increase in register switching energy. The energy-per-channel for the DSP, therefore, reaches a minimum at 8–16 channel

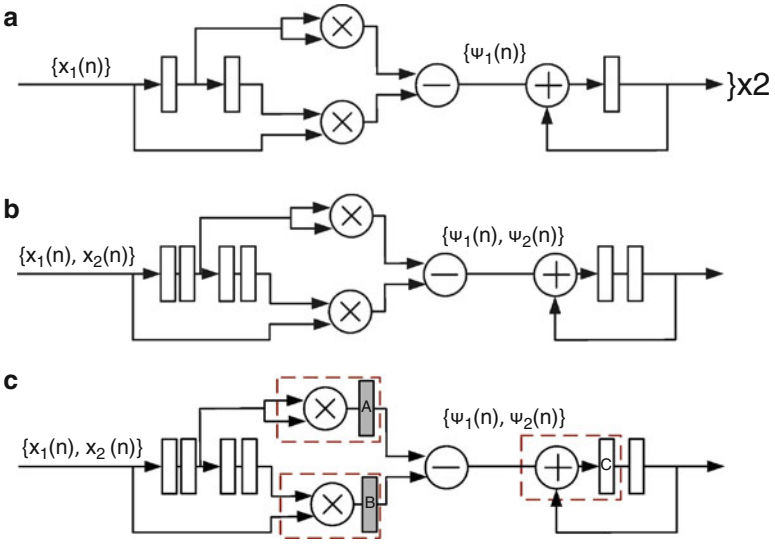

Fig. 2.9 Interleaved NEO detection logic

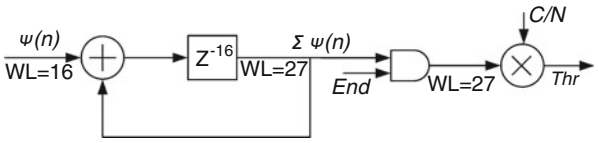

Fig. 2.10 Logic restructuring and wordlength optimization

interleaving. Thus, a good rule of thumb for spike-sorting DSP design is to interleave about 16 channels in the DSP core.

In addition to interleaving, logic restructuring and wordlength optimization can also be used to provide further power reduction. Attention should be paid to avoid redundant signal switching [23]. For example, consider the circuit shown in Fig. 2.10. This circuit is an accumulator, typically required for threshold calculation in various spike-sorting steps. To avoid redundant switching, the output of the accumulation node ($\Sigma\Psi(n)$) is gated such that the division for averaging happens only once, at the end of the training period, which is determined by the control signal *End*. This avoids redundant switching as $\Psi(n)$ is being accumulated. This strategy should be extended to the block level to ensure that the inputs to any given block switch only at the correct clock cycle. Using this method the switching activity of blocks that follow spike detection can be reduced by about 5× as the inputs to these blocks toggle only upon a spike detection event. Wordlength optimization can be performed using automated tools [24]. Iteratively relaxed (increasing MSE) constraints need to be specified on the mean squared error (MSE) at the signal of interest until detection or classification errors occur for a

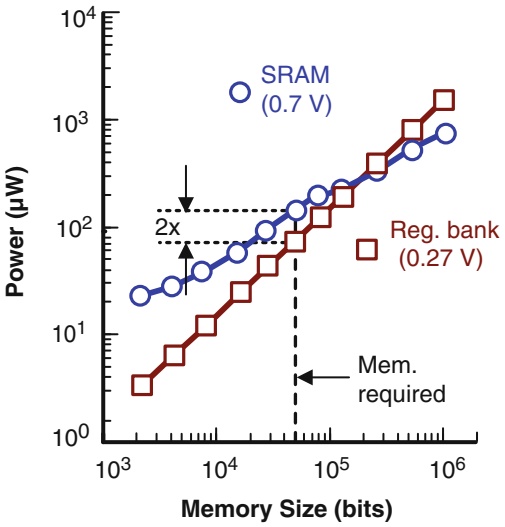

Fig. 2.11 Logic restructuring and wordlength optimization

set of the input test vectors. Wordlength optimization can offer an area reduction of up to 15 % compared to a fixed MSE design [22].

Memory architecture deserves special attention in spike-sorting DSP designs since memories often dominate their power consumption. The supply voltage for compiled commercial SRAMs cannot be scaled to below 700 mV due to their limited read noise margin. The supply voltage for register-bank memories, on the other hand, can reliably be scaled to subthreshold voltages to reduce their leakage power. Figure 2.11 compares the power consumption of register bank and SRAM memories for various memory sizes. For typical memory sizes of about 50 kb [20, 22] required for spike-sorting DSPs, the power consumption of register-bank memories is around 2 times lower than the power consumed by an equivalent SRAM. Thus, register-bank memories provide a more power-efficient alternative over conventional SRAMs. However, register-bank memories have significantly higher area compared to SRAMs. To overcome this limitation, specially designed subthreshold SRAM cells [25] could be used in spike-sorting DSPs.

Spike-sorting DSPs do not need random access to individual spike samples, as the algorithms perform operations on an entire spike waveform. Hence, the memories in spike-sorting DSPs can be organized as spike registers, as shown in Fig. 2.12. Each spike register is used to save a single spike waveform that is N_{spk} samples long. By organizing the memory as spike registers, the power consumed by the decoder is reduced by N_{spk} times. Each spike register module consists of 8-bit registers to save the spike waveforms and a delay line for clock gating. To select the spike from a particular spike register (Spike Reg. j), the decoder enables the clock to the spike register ($C\,l\,k_{enj}$) for one clock cycle. This enables the clock to access the first sample ($S(1)$) of the spike waveform. In the next clock cycle, the clock enable signal is shifted in position through the delay line to enable access to the second sample ($S(2)$) of the spike waveform. This process is repeated until all the samples of the spike

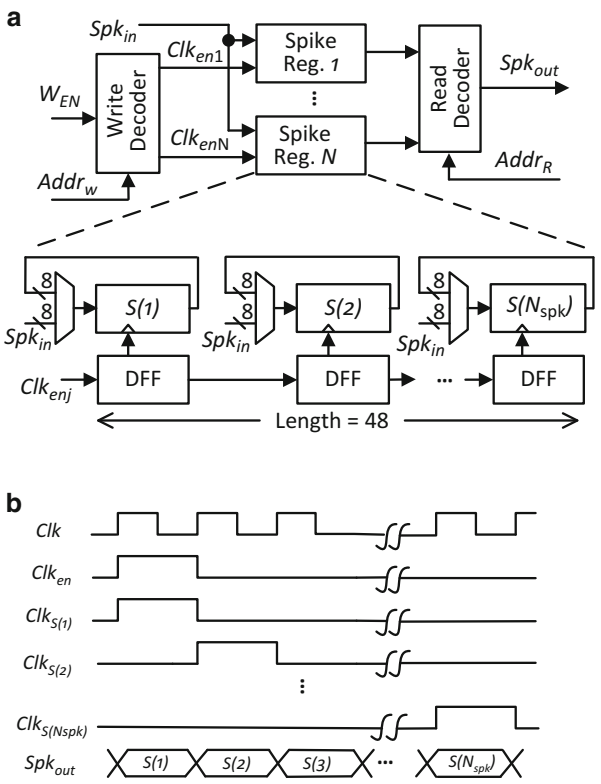

Fig. 2.12 Architecture of register-bank memories for spike-sorting DSPs

waveform have been accessed. In this architecture, only 1-bit D-flip-flops have an active clock, not the 8-bit registers. This delay-line-based clock-gating scheme, hence, reduces the power consumed in the redundant clock transitions by 8 times.

We have demonstrated the use of the above techniques to implement multi-channel spike-sorting DSPs with a power consumption of less than 5 μW/channel in [22] and [20].

2.4 Future Directions in Spike-Sorting DSP Design

This chapter serves as an introduction to spike sorting and compressed sensing for neural recording. We summarized some important algorithm-, architecture-, and circuit-level techniques that can be used for an efficient implementation of neural signal processing DSPs. Both spike sorting and CS provide output data-rate reduction that serves to reduce the total system power. Table 2.1 lists the typical data-rate reductions from spike sorting and compressive sampling. It should be noted that compressive sampling provides similar data-rate reduction as that provided by extracted features, but allows for accurate signal reconstruction at the receiver.

Table 2.1 Data-rate reduction for various output options

Output data	Data-rate reduction
Spike features	11x
Compressed samples	9.6x
Cluster IDs	240x

In addition to spike sorting, applications like brain–machine interfaces require robust decoding algorithms, whose implementation also needs to be investigated. Similar to spike-sorting DSPs, implementations of most biosignal processing algorithms are memory dominated. This raises the need for research on low-voltage, low-power memories for biosignal processors. While implementing a fixed set of algorithms may work well for clinical applications, flexibility in choosing the processing algorithms is a key requirement for neuroscientific applications. To this end, an exciting area of research is the development of Bio-FPGAs, i.e. FPGA implementations specifically tailored to suit the need of biosignal processing algorithms. All such developmental work needs to happen in a close, interdisciplinary collaboration with neuroscientists and clinicians.

Real-time processing of data is a key requirement for applications like brain–machine interfaces. However, many applications in neuroscience research do not require real-time processing. This being said, the recorded data easily occupies several terabytes of storage per day. Hence, data-rate reduction is a key requirement even for systems with off-line processing. The processing of this large volume of data in software is very time consuming and limits the productivity of research. Hardware accelerators for spike-sorting algorithms are required to speed up the processing of recorded data. Implementation of these accelerators would use highly parallelized architectures as opposed to the serialized architecture used for online spike sorting. Overall, the successful pursuit of such opportunities in the area of biosignal DSP design requires a significant contribution from the modern-day digital circuit designer.

References

1. J. Du, T.J. Blanche, R.R. Harrison, H.A. Lester, S.C. Masmanidis, Multiplexed, high density electrophysiology with nanofabricated neural probes. PLoS One **16**, e26204 (2011)
2. G. Santhanam, S. Ryu, B. Yu, A. Afshar, K. Shenoy, A high-performance brain-computer interface. Nature **442**, 195–198 (2006)
3. R. Quiroga, Z. Nadasdy, Y. Ben-Shaul, Unsupervised spike-detection and sorting with wavelets and superparamagnetic clustering. Neural Comput. **16**(8), 1661–1687 (2004)
4. Z. Nadasdy, et al., Comparison of unsupervised algorithms for on-line and off-line spike sorting, presented at the *32nd Annual Meeting of the Society for Neuroscience*, 2002, http://www.vis.caltech.edu/~zoltan/
5. M. Abeles, J. Goldstein, Multispike train analysis. Proc. IEEE **65**(5), 762–773 (1977)
6. A. Zviagintsev, Y. Perelman, R. Ginosar, Low-power architectures for spike sorting. In: *Proceedings of International Conference on Neural Engineering*, pp. 162–165, 2005
7. Q. Quiroga, Z. Nadasdy, Y. Ben-Shaul, Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural. Comp. **16**, 1661–1687 (2004)

8. U. Rutishauser, E.M. Schuman, A.N. Mamelak, Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. J. Neurosci. Meth. **154**, 204–224 (2006)
9. T.M. Sees, et al., Characterization of tissue morphology, angiogenesis, and temperature in adaptive response of muscle tissue to chronic heating. Lab. Investig. **78**(12) (1998)
10. M. Juusola, H. Robinson, G. de Plavieja Coding with spike shapes and graded potentials in cortical networks. Bioessays **29**(2), 178–187 (2010)
11. E. Candes, M. Wakin (2008) An introduction to compressive sampling. IEEE Signal Process. Mag. **25**(2), 21–30 (2008)
12. Z. Charbiwala, V. Karkare, S. Gibson, D. Markovic, M. Srivastava, Compressive sampling of neural action potentials using a learned union of supports. In: *Body Sensors Networks Conference*, May 2011
13. S. Chen, D. Donho, M. Saunders, Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**(1), 33–61 (1998)
14. E. Candes, T. Tao, Decoding by linear programming. IEEE Trans. Inf. Theory **51**(12), 4203–4215 (2005)
15. W. Lu, N. Vaswani, Modified compressive sensing for real-time dynamic MR imaging. *16th IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 3045–3048
16. M. Lewicki, A review of methods for spike sorting: the detection and classification of neural action potentials. Network Comput. Neural Syst. **9**(4), 53–78 (1998)
17. S. Gibson, J.W. Judy, D. Marković, Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. IEEE Trans. Neural Syst. Rehabil. Eng. **18**(4), 469–478 (2010)
18. S. Mukhopadhyay, G. Ray, A new interpretation of nonlinear energy operator and its efficacy in spike detection. IEEE Trans. Biomed. Eng. **45**, 180–187 (1998)
19. R. Chandra, L.M. Optican, Detection, classification, and superposition resolution of action potentials in multiunit single-channel recordings by an on-line real-time neural network. IEEE Trans. Biomed. Eng. **44**(5), 403–412 (1997)
20. V. Karkare, S. Gibson, C.-H. Yang, H. Chen, D. Marković, A 75W, 16-channel neural spike-sorting processor with unsupervised clustering. IEEE Symp. VLSI Circ. 252–253 (2011)
21. V. Karkare, S. Gibson, D. Marković, A 130 uW, 64-channel spike-sorting DSP chip. In: *IEEE Asian Solid-State Circuits Conference*, pp. 289–292, November 2009
22. V. Karkare, S. Gibson, D. Marković, A 130 μW, 64-Channel, Spike-Sorting DSP Chip. IEEE J. Solid State Circ. **46**(5), 1214–1222 (2011)
23. A.P. Chandrakasan, Low Power Digital CMOS Design. Ph.D. Thesis, University of California, Berkeley, Summer 1994
24. D. Marković, B. Nikolić, R.W. Brodersen, Power and area minimization for multidimensinal signal processing. IEEE J. Solid State Circ. **42**(4), 922–934 (2007)
25. B.H. Calhoun, A. Chandrakasan, A 256-kb 65-nm subthreshold SRAM design for ultra-low voltage operation. IEEE J. Solid State Circ. **42**(3), 680–688 (2007)

http://www.springer.com/978-1-4614-8150-8

Neural Computation, Neural Devices, and Neural Prosthesis

Yang, Z. (Ed.)
2014, X, 371 p. 144 illus., 102 illus. in color., Hardcover
ISBN: 978-1-4614-8150-8