Marple: Scalable Spike Sorting for Untethered Brain-Machine Interfacing

Eugene Sha
University of Toronto
Toronto, Canada
eugene.sha@mail.utoronto.ca

Andy Liu
University of Toronto
Toronto, Canada
andyw.liu@mail.utoronto.ca

Kareem Ibrahim
University of Toronto
Toronto, Canada
kareem.ibrahim@mail.utoronto.ca

Mostafa Mahmoud
University of Toronto
Toronto, Canada
mostafa.mahmoud@mail.utoronto.ca

Christina Giannoula
University of Toronto
Toronto, Canada
christina.giann@gmail.com

Ameer Abdelhadi
McMaster University
Hamilton, Canada
ameer@mcmaster.ca

Andreas Moshovos
University of Toronto & Vector Institute
Toronto, Canada
moshovos@eecg.toronto.edu

Abstract

Spike sorting is the process of parsing electrophysiological signals from neurons to identify if, when, and which particular neurons fire. Spike sorting is a particularly difficult task in computational neuroscience due to the growing scale of recording technologies and complexity in traditional spike sorting algorithms. Previous spike sorters can be divided into software-based and hardware-based solutions. Software solutions are highly accurate but operate on recordings after-the-fact, and often require utilization of high-power GPUs to process in a timely fashion, and they cannot be used in portable applications. Hardware solutions suffer in terms of accuracy due to the simplification of mechanisms for implementation's sake and process only up to 128 inputs. This work answers the question: "How much computation power and memory storage is needed to sort spikes from 1000s of channels to keep up with advances in probe technology?" We analyze the computational and memory requirements for modern software spike sorters to identify their potential bottlenecks - namely in the template memory storage. We architect *Marple*, a highly optimized hardware pipeline for spike sorting which incorporates a novel mechanism to reduce the template memory storage from 8 - 11x. Marple is scalable, uses a flexible vector-based back-end to perform neuron identification, and a fixed-function front-end to filter the incoming streams into areas of interest. The implementation is projected to use just 79mW in 7nm, when spike sorting 10K channels at peak activity. We further demonstrate, for the first time, a machine learning replacement for the template matching stage.

CCS Concepts: • **Hardware → Neural systems**; *Hardware-software codesign*; Hardware accelerators.

Keywords: Spike Sorting,Computational Neuroscience

ACM Reference Format:
Eugene Sha, Andy Liu, Kareem Ibrahim, Mostafa Mahmoud, Christina Giannoula, Ameer Abdelhadi, and Andreas Moshovos. 2024. Marple: Scalable Spike Sorting for Untethered Brain-Machine Interfacing. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3620665.3640357

ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA

ACM ISBN 979-8-4007-0385-0/24/04
https://doi.org/10.1145/3620665.3640357

1 Introduction

The human brain comprises billions of neurons that communicate through electrophysiological signals called *spikes*, which serve as the fundamental units of brain communication. To better understand complex brain behaviors and structures, neuroscientists employ *spike sorting*, a process that attributes spikes to their respective firing neurons. This single-neuron activity reveals higher-order brain functionality [3, 19, 20, 62]. Real-time interaction via neuronal communication enables life-changing advances, e.g. motor control for paralysis patients [22, 42], epilepsy detection and mitigation [14, 44, 85], treatment of Parkinson's disease [29, 33], and cognitive control [77]. These *early* successes are riding on a sustained wave of *exponential* growth [79] in electrode count and continues unabated, with implantable neural

probes having several thousands of electrodes being commercially available [78] and prototypes with even more electrodes underway [67]. Apart from the existing applications, larger scale ones remain unrealized due to the many impediments to perform spike sorting at a high-scale (see Section 2). There are three foundational technologies that need to scale to more than tens of thousands of neurons for such applications to begin to materialize: 1) implantable voltage sensors, 2) an analog-to-digital front-end voltage converter, and 3) a digital processing back-end (the focus of this work). Today, implantable probes and analog-to-digital conversion have already reached such scales [52, 67, 78] and continue to outpace the digital back-end. Therefore, realizing the potential of such brain-machine applications hinges upon the digital back-end 1) to observe and act upon activity across orders of magnitude more neurons, and 2) to do so in real-time using wearable, energy-efficient systems that operate autonomously for long periods of time [40, 75].

Near-brain implants, including neural prosthetics and brain-machine interfaces (BMIs), should incorporate a well-defined power budget due to two compelling reasons supported by scientific research and clinical considerations. Firstly, safety is a crucial concern for many near-brain implants. Excessive power consumption can lead to heat generation, potentially damaging sensitive brain tissue. Therefore, a power budget helps ensure that the implant operates within safe temperature limits, safeguarding the well-being of the patient [66]. Battery life is another paramount concern. These devices often rely on batteries for power, and optimizing power usage is essential to prolong battery life. Longer battery life reduces the frequency of recharging batteries, enhancing the patient's quality of life and reducing associated risks [34]. The power budget of near-brain implants can vary significantly depending on the specific device, its intended application, and technological advancements. However, recent scientific findings and clinical considerations strongly suggest that a power budget of 2W should be considered [32, 47, 75]. Existing commodity processing systems today, e.g., CPUs and GPUs, are far from capable of meeting the stringent combination of processing capability and power efficiency needed to keep up with these advances (See Section 3.2). To fill this gap, a few custom-built systems have been developed only at low-scales [30, 92]. Presently, development of potential applications with even a few hundreds of neurons requires tethering to a server and offline analysis [25] which severely limits their utility.

Fully-implantable devices illustrate the inherent challenges in device design due to their stringent constraints. In addition to the portability of a near-brain implant, fully-implantable devices are restricted to smaller form factors, have durability and longevity considerations [61], and must adhere to stricter power budgets due to the 1°C thermal safety threshold of the International Organization for Standardization to prevent brain damage and cell death [24]. The power budget is limited to 47-81 mW ([73]), but this can reduce further depending on the device's spatial footprint. Current technology scales to only hundreds of channels within a 50 mW power budget ([32, 51]). Even for simple components, scaling to ten thousand channels and beyond will require great innovation. Current neural amplifiers consume 0.5-10 μW per channel [17]. For ten thousand channels, amplifiers alone consume 5-100 mW, surpassing the entire power budget.

Therefore, the goals of our work at large are 1) to investigate the needs of such applications as the input neuron count scales up in the thousands, and 2) to architect an appropriate system to best serve them. A key challenge with this endeavour is that the domain specific applications are not mature and well defined. Regardless, meaningful progress can still be made. The prevailing consensus is that an essential processing step for many such applications is *spike sorting*. Its indispensability has motivated decades of continuing research and development, e.g., [6, 7, 35, 36, 53, 59, 63], the establishment of comprehensive software implementations [10, 54, 94], and of benchmarking resources and methodologies [41]. To complement and build upon these prior works, the first goal of this work is - for the first time - to quantify what it takes to perform state-of-the-art spike sorting in real-time for thousands of probe channels and in a wearable form factor. Our second goal is to architect a hardware pipeline to enable spike sorting at the scale of tens of thousands of neurons.

We begin by analyzing state-of-the-art software spike sorting pipelines which use *template matching*, the most effective and mature to-date method [10, 41]. Template matching uses a set of prerecorded spike waveforms, comparing against inputs to identify the source neuron. Our first contribution is the modeling and analysis of its computation and memory needs as a function of input channel count. The analysis shows that software-based implementation cannot scale up to thousands of neurons with memory and computation needs far exceeding of even high-end processing cores. Indicatively, keeping pace with the input stream from 10K channels requires > 100B instructions/second out of which 75B is solely to identify windows of interest where spikes may be occurring. This is challenging even for high-end CPUs and GPUs, let alone for a wearable, energy efficient system. Memory demands are also problematic for scaling as they reach 16M elements for template storage alone.

This analysis motivates our second contribution, *Marple*, a custom hardware architecture for high-channel count spike sorting – we evaluate a system of up to 10K channels or 30K neurons in wearable applications. Marple has two major components: The first is a series of fixed-logic processing stages which aims to denoise input waveforms and to identify areas of interest. These are spatiotemporal windows into channel streams which may contain a spike. Each window is centered around a local peak in the input signals and contains samples around the peak from all relevant neighboring channels. The second component performs the *template matching*

Table 1. Terminology

Term	Meaning
Probe	An invasive implantable device used to record electrophysiological signals from the brain (Figure 3). Also known as neural probe.
Channel	A recording site of a probe. Also referred to as an electrode.
Density	With respect to probes, density refers to the number of channels on a single probe. A higher density means a higher channel count.
Pitch	Distance between two adjacent channels.
Sample	A voltage reading from a channel at a given time.
Spike	The sequence of samples signalling a neuron's activation, typically 1-2ms in duration.
Morphology	The shape of a spike which has particular characteristics (Figure 2).
Cluster	A group or set of N-dimensional points, often in the context of sorting or classification. An example is a collection of spikes belonging to a particular neuron that are part of the same cluster.
Template	A proxy of a neuron's spike, identified by clustering. Typically, this is the centroid of a cluster.

step, where the window is compared against prerecorded templates in order to identify the source neuron. Our design uses a flexible vector processing unit to perform the template matching. The key innovation is a lightweight template compression method that makes it practical to store the templates. Marple's two component architecture is flexible and broadly useful: The front-end, "window of interest" unit can be used with other back-end spike identification methods. To illustrate this flexibility, we present a novel, machine-learning back-end which uses a neural network to identify the source neuron, given an input window of interest. Our vector-based back-end can directly execute the model. However, further optimization is needed for this method to meet real-time constraints. We highlight the following findings: 1) Our analytical model (derived from SOTA spike sorters [10, 54, 94]) finds that for high scales (30k neurons), spike templates take up to 90% of the overall storage requirements. 2) Our template compression method reduces template storage by 8 - 11x while retaining +99% relative accuracy to a high-performance spike sorter [10]. 3) Our design and implementation of a high-performance online spike sorter in hardware, providing power and area estimates for large-scale workloads. Our design can sustain peak processing for 30K neurons, consuming only 78.08mW (post-layout measurements scaled from 65nm to 7nm).

2 Background and Motivation

In a complete BMI system, neural signals must be collected, correctly attributed, interpreted, and then acted upon to induce a desirable effect. A BMI system is composed of a sensory input, analog data acquisition, and a digital computing stack composed of a spike sorter and an activity decoder. The pipeline is depicted in Figure 1. Table 1 lists relevant terminology used in spike sorting and throughout this paper. The inputs to spike sorters are electrophysiological signals from neural probes. Neural probes are invasive implants that record, amplify and digitize voltages produced by neurons into streams. Modern probes have channel layouts which can vary from linear shanks, 2D grids, to 3D matrices (see Figure 3). As the probes increase in density, the pitch *can* decreases to the micron range. Due to the proximity, spikes are often recorded on multiple nearby channels and provide spatial information. The key aspects of probe design that influence the computations downstream are the sampling rate, bitrate, number of channels, and layout. Sampling rates are commonly around 30 kHz and bitrates around 10-16 bits per sample [27, 49, 58, 78, 86]. The number of channels currently ranges upwards of tens of thousands [58, 67, 78, 95] and over time has shown exponential growth [79], necessitating improvements to software and hardware designs. The digital compute stack consists of 1) a spike sorter which aims to match each detected spike to the corresponding neuron that generated it and 2) an activity decoder that deciphers the brain activity when reading groups of spikes.

2.1 On the Necessity of Large-Scale Spike Sorters

Apart from the existing applications of spike sorting, including epilepsy detection and mitigation [14, 44, 85], treatment of Parkinson's disease [29, 33], and cognitive control [77], larger scale applications remain unrealized due to the many stringent requirements to perform spike sorting at a high scale. Traditional spike sorters are not capable of keeping pace with the exponential growth in incoming data [79], requiring massively more computation and memory [6, 7, 35, 63]. Spike sorters have also seen drastic increases in algorithmic complexity [88], with further area and power constraints vital to advancements of untethered applications [40, 75]. The promise of such applications has been fueling a sustained wave of *exponential* growth [79] in probe technology that continues unabated; probes containing thousands of electrodes (channels) commercially available [78] and prototypes with even more electrodes underway [67]. At the same time, advances in the analog front-end have also kept pace, e.g., [52, 67, 78]. However, commodity systems existing today cannot meet the constraints for latency and portability for keeping up with these advances (section 3.2). To fill this gap a few custom-built systems have been developed [30, 92]. Presently, development of potential applications with even a few hundreds of neurons requires tethering to a server and offline analysis [25].

In the past decade, there has been an influx of new spike sorters [10, 26, 38, 39, 53, 54, 60, 68, 71, 87], but current solutions have shortcomings in one of four ways: 1) They do not operate in real-time [38, 68]. 2) They are not accurate enough [9]. 3) They are not portable [10, 26, 53, 54, 94] as many are software solutions. 4) For hardware solutions, they do not operate at the scale of modern probe technologies [71, 87, 93], requiring more efficient implementations for online spike sorting, especially if implantable BMIs are to be portable and responsive. The aforementioned spike sorters deal with tens and hundreds of neurons, with some accurate enough to scale up to thousands. However, there

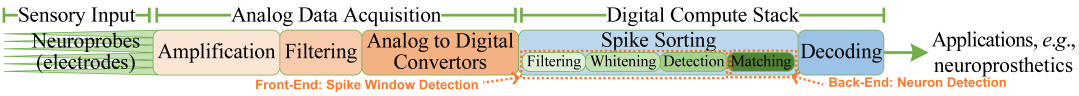

Figure 1. A brain-machine interface pipeline.

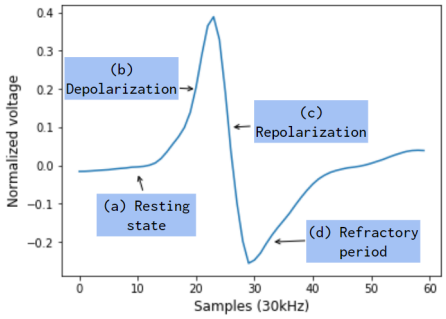

Figure 2. The common stages of a neuronal action potential: the resting state (a), depolarization (b), repolarization (c) then reaching a refractory period (d) before returning to (a).

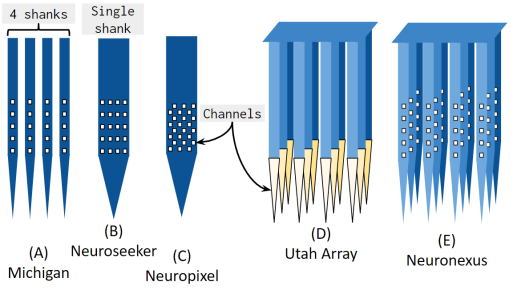

Figure 3. Common neural probe topologies. (A) quad-shank Michigan probe [90] (B) NeuroSeeker probe [18] (C) Neuropixel probe [58] (D) Utah array [42] (E) 3D NeuroNexus matrix [50]

are some key limitations. Most software spike sorters process *offline* after a recording has been stored, which by its nature limits the responsiveness and portability of the software, as it runs on modern desktop and server class systems. Kilosort [53, 54, 78] is a modern spike sorter that can run online after calibration but must use a desktop-class GPU to achieve *real-time* performance, and thus is not portable.

For BMIs to operate on thousands of neurons, the spike sorter must satisfy the following requirements: i) perform on-the-fly processing at real-time latency, ii) low area and power energy costs for portability, and iii) scaling to processing thousands of neurons very accurately. The processing must be done on-the-fly to be responsive, with a tight real-time latency budget (e.g., $<$ 50ms for closed-loop manipulation [12]). The area cost as well has to be considered, as desktop-class systems are not portable. Energy and power consumption must also be considered with untethered applications constrained to $<$ 2W [47, 75] for portability and potential implantation. Spike sorters must also scale to keep pace with exponential growth of the analog front end [7, 63].

2.2 On the Applicability of Spike Sorting

While the neuroscience community has made great strides in improving spike sorting on multiple fronts, there are a few proponents that challenge its necessity [70, 82, 83, 89]. Ventura et al. [89] demonstrate an alternative method to decode spikes without traditional spike sorters. However, their method was only tested up to forty electrodes and suffered from implementation issues, taking 10-20 seconds to process each electrode. An analysis by Todorova et al. [82] demonstrates the advantage of using spike sorting in a spike decoding process. They specifically isolate and vary the processing prior to spike decoding, finding that discarding unattributed threshold crossings degrades the downstream decoding, and ultimately concluding that "spike-sorting is useful".

The utility of spike sorting is often questioned for applications which make use of population-level dynamics [11, 37, 55, 74, 83]. For example, motor control has been analyzed using population dynamics with [81] and without [11] spike sorting. However, the availability of non-invasive technologies should not preclude the use of spike sorting, and in fact can be used in tandem to great effect [91]. For example, in a study of memory retrieval and delayed learning relating to adverse effects from novel foods, the authors first broadly locate areas of interest with whole-brain light sheet imaging, then use spike sorting to identify the neurons responsible for memory retrieval (for novel flavors) and delayed learning (if the food lead to discomfort) [97]. They construct three hypotheses by targeting novel flavor-coding neurons (NFC) and calcitonin gene-related peptide neurons (CGRP, malaise detecting): 1) Individual NFCs stay active, overlapping with the eventual activity from CGRPs. 2) CGRPs specifically reactivate the NFCs. 3) CGRPs activate a new, separate group of neurons (indicative of malaise), which then becomes associated with the NFCs in future memory retrievals. The unit-level precision of spike sorting enabled the researchers to confirm the second of the three hypotheses.

For applications employing single-unit activity (e.g. neural stimulation, and the understanding of visual coding, behaviours, and neuronal circuitry [4]), spike sorting has no clear substitute. In a study of ocular dominance, spike sorting was necessary to understand the impact of left-right preference weightings of individual neurons in the visual

cortex [45]. Spike sorting is vital for thorough characterization of neuronal circuitry. E.g. in understanding thirst motivation [1] where an analysis of 23,881 neurons across 34 regions of the brain, recorded over 87 sessions was necessary, an undertaking where a scalable online spike sorter would drastically reduce the manual effort required to conduct such studies. The complexities of neuronal memory elements also require spike sorting. E.g. as performed in the primary motor cortex [76], spike sorting was employed to uncover the individual L5a neurons used to retain information. While these question remains, we should note that the findings against the use of spike sorting are few and far between, implying the continued utility of spike sorting in modern times.

3 Spike Sorting in Traditional Architectures

The goal of spike sorting is to discern *when* and *which* neuron "fires" given the raw output from the analog front-end. More formally, spike sorting is a *source separation process* [35, 36, 63] which aims to attribute the recorded spikes to individual neurons, while separating background activity from local field potentials and noise (e.g. recording artifacts). This is challenging for several reasons: 1) While morphologically spikes look similar across neurons, their actual shapes vary in time, with the probe's placement, and by the neuron itself. 2) A channel can sense the superimposition of activity from many "nearby" neurons, as well as background activities in the brain. 3) Due to the lack of large in-vivo datasets, there is often no ground truth to appropriately determine accuracy. These factors jointly obfuscate the process, as it is difficult to discern whether similar spikes across nearby channels are from a single neuron or multiple. The challenges are addressed by: 1) An active research effort to improve spike sorting algorithms (and with it, a growing complexity) [88]. 2) Decades of neural experiments, culminating in the modern understanding of the foundational biophysics in the brain. This directly informs 3) the generation of synthetic datasets from the corpus of live cell models to provide ground truth data for objective and equal benchmarking [5, 8, 21, 41].

3.1 Stages of online spike sorting

Figure 4 shows a typical state-of-the-art *online* spike sorting pipeline [10, 54, 94], with the flow of data and compute. Spike sorting can be performed offline after the full recording is available. however, we target online processing utilizing spike templates that have been derived through prior offline runs as this is desirable for quick feedback [12] and portability [75]. Calibrating the templates offline is used to tune to each subject and each application. Our online pipeline is modeled after SpykingCircus [94] and Kilosort [54], with templates generated offline via MountainSort4 [10].

Bandpass filtering: The incoming signals contain unwanted local field potentials at the lower frequencies (<100-300 Hz) and high frequency noise (>3-6 kHz) which the first stage filters out. We assume the use of the 3rd order Butterworth filter due to its widespread usage [10, 32, 49, 54, 64, 71, 78, 94]. Bandpass filtering occurs for every channel independently, scaling linearly with channel count.

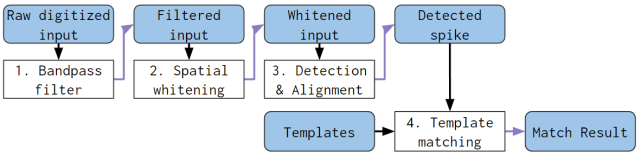

Figure 4. The stages of an online spike sorter. White boxes are functional stages. Blue boxes are the input & output data.

Whitening: After temporal noise is filtered, *whitening* removes spatially correlated noise from neurons that affect a large area, but are too far to be distinguished [10, 54, 94]. Every channel has a whitening matrix derived from the covariance matrix of regions of silence. We opt for *local whitening* where only nearby channels contribute to the covariance matrix, capping its total size to $C \times C_{tr}$ where $C_{tr} \ll C$, and C the total channel count (global whitening is unnecessary due to negligible spatial noise from distant channels).

Detection: The denoised activity is checked for spikes (i.e. *if* a neuron has fired). The defacto approach uses an unsupervised threshold $Thr = 4\sigma$ [48, 56, 59, 68], where $\sigma = med\{\frac{|x|}{0.6745}\}$ and x is a long (e.g. 30 second) input stream for a channel. Thr is an estimate based on the median of the filtered signal which acts as a proxy for the standard deviation of noise. Signals crossing the threshold are classified as spikes. Simpler methods, such as the nonlinear energy operator (NEO) detection method, achieve similar accuracy for small datasets [30, 93]. More complex detection methods, such as deep neural networks [69] are promising but incur significant compute and memory costs.

Alignment: Downstream classification requires a *window* of samples centered at the trigger that is *aligned* at the peak amplitude [48, 59, 87]. The spike's duration vary [84], but 2ms holds as a consensus [26, 59, 72] (60 samples at a 30 kHz sampling rate). Spikes are often detected in a neighborhood of nearby channels with the maximum amplitude assigned as the central channel [10, 54, 94]. A neighborhood provides spatiotemporal information (e.g., the relative amplitude and delay in sensing the trigger), improving classification accuracy. We use neighborhoods of the 9 closest channels, and a maximum time difference of 10 samples between channels to account for intra-neighborhood delays [10].

Template matching: The final stage performs a vector dot-product of the input spike (9 channels by 60 samples) against pre-calibrated templates (same size as the spike) to produce a correlation score. The maximum correlation exceeding a threshold is detected as the source unit. The number of templates to compare against varies per channel. For our

datasets, the maximum is 13 templates per channel, with an average of 3 and standard deviation of 2.3.

Offline Template Generation: The templates needed by the final stage of the online spike sorter are generated offline when calibrating the probe(s). Clustering divides the spikes into groups, where each group encloses spikes of a neuron. This is an unsupervised process where the number of neurons is not known beforehand. Generally, templates are the centroids of each cluster and approximate a neuron's spike. Templates are attributed to a single *central* channel where they have the strongest signal, while capturing the spike over the neighborhood of channels. This can change over time due to drift [6], although calibration of templates can resolve this. We use MountainSort4 to generate templates [10].

3.2 System-Level Needs of Large-Scale Spike Sorting

This work studies the scalability of the SS pipeline to higher neuron counts and examines the compute and storage costs associated with varying firing rates and neuron counts. Table 3 analytically models the memory footprint and computation costs of online SS. Figure 5 reports how these costs scale as a function of channel count C and firing rate F (in spikes/sec/neuron). Both parameters have nearly linear effects on costs. F has a minor effect compared to the dominant C. The computation and memory costs for C=100 are minimal, suggesting that even a software implementation with minor hardware assists may be sufficient and preferable for flexibility. For our experiments, we consider a high channel count scenario C=10K neurons which fire at 5Hz per neuron, leading to a total computation cost of 25 GOPs. We also investigate the impact of a higher firing rate of 20Hz, which increases the computation costs to 41 GOPs.

CPU: To address the computational requirements, we implemented an optimized spike sorting pipeline in C, compiled for an i9-9900K CPU using *gcc* with -O3 optimization and AVX extensions. For a slow firing rate of 5Hz and an average of 3 templates per channel, the program needs to execute 107B $\frac{instructions}{second}$, which scales up to 145B $\frac{instructions}{second}$ for a 20Hz firing rate. The CPU's power consumption of 95W alone makes it impractical for standard workloads [23]. On the other hand, lower power processors are not able to handle the required instruction count for the pipeline.

GPU: Modern GPUs are likely to meet the processing requirements but their power consumption is prohibitive. We follow the approach of Arafa et al. [2] and analyze the case of 41 GOPs of compute. Our findings indicate that an average of 57.4W is required to maintain the desired firing rate, which exceeds our power constraints. Even the most recently *announced* low power GPUs such as NVIDIA's Jetson Orin Nano, which consumes **between 7 and 10 watts of power** under typical workloads [13], still far exceed the application's power constraints.

We propose a solution with significantly reduced power consumption which is **less than 0.1W** when scaled to the recent technology nodes, as shown in Table 7. This highlights the potential for more power-efficient hardware solutions for SS pipelines with high neuron counts. In addition to the aforementioned challenges, we must also address the costs associated with memory requirements, particularly for template storage. With C=10K and F=20, 16.2 million single-precision floating-point values (FP32) are needed for template storage, making up 90.4% of the total memory costs. While 65MB is feasible for SRAM or DRAM, it is not ideal for untethered applications due to the high energy costs with the random access of templates during matching.

Table 2. Instruction Counts for Spike Sorting on a CPU.

Unit	Stage	Insts./Second (B)
Front-End	**Filtering**	25
	Whitening	70
	Spike Detection	0.08
Back-End	**Template Matching (F=5)**	12.8
	Template Matching (F=20)	50.8
	Total (F=5)	107.88
	Total (F=20)	145.88

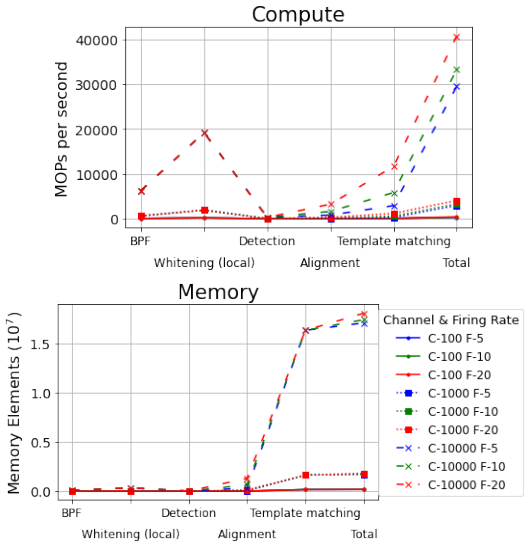

Figure 5. The compute (top) and memory (bottom) costs for the online spike sorting pipeline. Scaling with the number of channels C and firing rate F.

4 Marple: Compression of Templates

We investigate template compression to extend the range of channel counts that can be practically processed in untethered applications. Templates are structured in three dimensions: scale, time and space. The *scale* is proportional to the number of neurons within the probe's detectable range, and grows linearly with channel count. *Time* is the number of samples in a template, proportional to a probe's sampling rate and spike width. *Space* is the neighborhood size.

Table 3. (Left) Analytical Model of Computation and Memory Element Counts. (Right) Definitions.

Stages	Compute [Operation Type]	Memory
BPF	$C \times S_R \times 11$ [MAC]	$10 + 8 \times C$
Whitening (local)	$32 \times C \times S_R$ [MAC]	$34 \times C$
Detection	$C \times S_R$ [Comparison]	C
Alignment	$N_b \times S_R \times F$ [Comparison]	$2 \times F \times S \times N_b \times S_R$
Template Matching	$T \times F \times$ $N_b \times S \times (T+1)$ [MAC & SUB] $+(T+2)$ [Comparison]	$N_b \times C \times \{(S+1) \times T\}$ $+2 \times T$

Parameter	Definition
C	Number of channels
S	Spike width (# samples)
S_R	Sampling rate
N_b	Neighborhood size (# channels)
F	Activity factor (# spikes/sec/neuron)
T	Templates per channel

Datasets: In the past, manual datasets have been the primary source for assessing the performance of spike sorters. These datasets are often collected from *juxtacellular* recordings, where a probe is placed both internally for exact spiking information, and externally for validation data to mimic settings that are not privy to the internal data (as in practical applications). However, this is a very costly process in time and effort, requiring an expert to deftly insert an electrode into individual cells — an impractical approach for more than handfuls of data points. For decades, the neuroscience community has turned to synthetic generation of cell recordings for evaluation as a proxy with ground truth data [8], with continual innovation in the frameworks used [5].

We employ two separate datasets to evaluate scalability. SpikeForest's (SF) datasets [41] provides manual, synthetic and hybrid recordings, ranging from single-neuron and single-channel recordings up to 708 neurons and 64 channels. We use recordings with a minimum of 10 neurons and 4 channels from 7 study sets **composed of 29 studies or 87 recordings**; many solutions exist for 1 and 4 channel counts —typically these considered data from a single study— which are not the focus of this work [16, 38, 60, 87]. To test for high scales, we generate recordings with a Neuropixel probe (NP) [58] using the standard MEArec flow [5]. This NP dataset contains twenty 30-second recordings with 384 channels and 1500 neurons each. We combine the NP datasets in three configurations: 1500, 10,500 and 30,000 neurons or 1, 7 and 20 NP probes, respectively.

Templates: The templates are the inputs to compression and are matrices of 60×9 (samples x channels around a center) FP32 values. We will refer to the 60 samples per channel as a *waveform*, as in there are 9 waveforms to a template.

Metrics: We define *accuracy* as the ratio of matching labels produced before (ground truth) and after compression (predictions). To quantify the memory costs and savings, we introduce the metric *bits-per-value* (BPV) which is agnostic to the size of the dataset, and amortizes the memory cost.

$$BPV = \frac{\text{Templates bits} + \text{Metadata overhead bits}}{\text{Number of template values}}$$

The baseline assumes FP32 values with no metadata overhead ($BPV = 32$). The following methods aim to maximize compression (minimize BPV) while accounting for overheads.

Reducing Template Footprint: Given the goals of portability and real-time performance, a lightweight low-energy decompression method is essential (compression is performed offline). We take advantage of 3 forms of similarity in neural signals: 1) similarities in relative dynamic ranges, 2) spatially across templates, 3) temporally within a template, along with other helpful optimizations to reduce the BPV.

Differences from Centroid Waveforms: To take advantage of 1), we use quantization to express values as fixed-point indices to a codebook. We further reduce the value ranges by taking advantage of 2), employing K-means clustering to find *whole* centroid waveforms (60 values in time from one channel). Waveforms of a cluster are represented as per sample *differences* from their centroid. It is those differences that we quantize into a codebook, as they have a significantly reduced dynamic range. We investigated several quantization methods, each separating values into outliers and non-outliers [96]. Outliers exceed a threshold magnitude and are stored in FP32, whereas non-outliers are binned and replaced with a short index. This index points to the representative value stored in a codebook. The two best methods are 1) *Linear Quantization Enhanced* (LQE) which evenly divides the value range into bins, using the mean of the values in each bin, and 2) *Hierarchical Agglomerative Clustering* (HAC) [28, 43] which evenly distributes values so bins contain a similar number of values, giving more fidelity for high-density ranges. Figure 6b reports relative accuracy vs. the resulting BPV.

Segmented Delta Encoding: We take advantage of the temporal similarities within the spikes by encoding consecutive *indices* as *deltas* (Δ), where the first value is a *base*. Rather than using a fixed number of bits for all indices (e.g., 5b for a 32 entry codebook), we use only as many bits as necessary (recorded in a metadata field) removing any prefix of 0 bits. Delta encoding suffers when the spike transitions from/to the resting periods as the waveform exhibits abrupt changes in magnitude as per Figure 2. We propose *Segmented Delta Encoding (SDE)* which splits waveforms into multiple

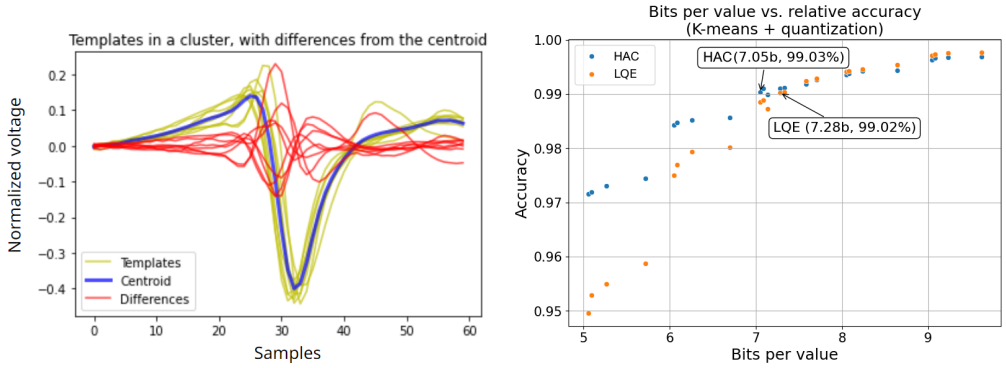

Figure 6. (Left): Waveforms in a K-means clusters (the templates, the centroid, and their differences to the centroid). Note the reduction in the range of values for differences. (Right): An evaluation sweeping configurations of codebook sizes and outlier thresholds for K-means clustering and indirect quantization against their accuracies on the SpikeForest datasets.

even segments, each encoded with its own base index. This is inspired by Base Delta Immediate (BDI) encoding [57], with two key differences. First, the bases themselves add no overhead as they are the first value of each segment (only the metadata to track the new length of Δ adds overhead e.g. 3b per segment to track lengths from [0, 7]). Second, our delta encoding is fundamentally different, as it is calculated as *consecutive* differences rather than as a difference from a fixed base. This accounts for +30% greater compression on average than BDI. Under our multiple workloads, we find that 6 and 10 segments performed the best for LQE and HAC.

Mantissa and Exponent Field Trimming: We trim the exponents and mantissas to reduce overheads from overprovisioning by the FP32 format. Exponents can be losslessly trimmed to 2 and 4 bits for the outlier and centroid values, respectively, down from 8 bits while their mantissas can be lossily trimmed to 4 and 7 bits in order to retain an average accuracy above 99% (a negligible loss of 0.01%). Outliers are therefore reduced to 7 bits (1 sign, 2 exponent, 4 mantissa) and centroids to 12 bits (1 sign, 4 exponent, 7 mantissa).

Duplicate segment storage: Representing the original waveforms as shorter sequences of Δ often results in duplicates amongst these sequences. With a larger number of segments, the shorter sequences are more likely to be duplicates (only 11% at 10-segments are unique in the NP dataset, compared to 65% in the SF dataset). We encode duplicates in a lookup table, and add a 1-bit duplicate flag, reusing the length and Δ fields in the templates as pointers to the lookup table. We sweep the pointer sizes and bit-lengths to find the optimal setting (9b pointers for 2b Δ).

Memory Footprint Reduction: Table 4 summarizes the reductions in BPV for each of the stages. Figure 7 reports the effect of template compression on overall memory footprint. Our compression methods reduce overall memory footprint to 1.4MB, a 5.7x reduction over the baseline.

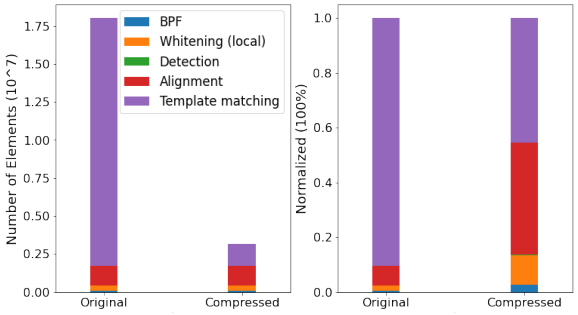

Figure 7. The memory cost of an online spike sorter for 10K channels and 20Hz firing rate from our analytical model. (Left) the absolute difference, (Right) the normalized memory costs. Template compression reduces memory from 18MB to 1.4MB, an overall reduction of 5.7x.

Table 4. BPV compression maintaining +99% accuracy. LQE outperforms HAC when all methods are applied, reducing the FP32 values to 2.83 BPV.

	Compression Method (Bits per value)			
Dataset	Template Centroids & Quantize	SDE	Datatype Trimming	Duplicate Dictionary
SF-HAC	7.05	5.73	5.00	4.99
SF-LQE	7.28	4.78	3.94	3.93
NP-HAC	5.32	3.61	3.42	2.92
NP-LQE	5.64	3.33	3.06	2.83

5 Marple: Architectural Design

The design of Marple, the online spike sorter is shown in Figure 8. Data flows between stages either directly or via scratchpad memories. We optimize Marple's organization by utilizing the input sample flow from the analog front end. Figure 11 illustrates the digitization process: data is converted into Q-bit (up to 16-bit) format using ADCs and

then serialized across C channels. Consequently, the spike sorting pipeline processes a single 16-bit sample per cycle. For a standard sampling rate of $f_s = 30\ kHz$ and a desired channel count of $C = 10K$, achieving real-time feedback necessitates an operational frequency of $f_{op} = 300\ MHz$.

5.1 The Marple Spike Sorter

The first stage performs digital *filtering* over the samples of a single channel. The filtered samples go through the Neighborhood Buffer which enables the second *Whitening* to seamlessly operate on samples from a neighborhood of channels. The remaining stages identify *where* (channel) and *when* spikes occur and present a window of 60 samples per channel over a neighborhood of 3×3 channels to the template matching stage so that it can identify the source neuron.

Filtering: We implement a 3rd order Butterworth IIR band-pass filter with a cascaded biquads implementation. For each FP32 output, the filter performs 12 multiplications plus 11 additions over what is effectively a 6 sample window. Given the relatively low sampling rate (30 kHz) time-domain multiplexing multiple channels to a filter reduces costs. A single set of multipliers and adders is sufficient as we time-multiplex them over the channels via a 10K scratchpad (one row per channel). Each row contains 6 FP32 values enabling a 6-stage pipelined filter implementation. Each cycle, we read one row and write another. The scratchpad is banked and since we process channels round-robin, each bank is single ported. Every cycle, this stage produces a single FP32 sample.

Neighborhood Buffer: Whitening operates over the samples of a group of channels. As previously seen, the channels of a probe are typically arranged in a uniform grid, which we denote as $P_W \times P_H$. From a central channel, its neighbors are channels within a distance of N_r (neighborhood radius). Figure 9 shows an example of a 7×6 channel probe and a neighborhood centered at channel 19 with a radius of $N_r = 2$. Whitening samples from channel 19 needs samples from the whole neighborhood in the same time frame. If the incoming data is organized line-by-line in memory, multiple read ports would be required as reading a whole neighborhood requires buffering until all channels are read. Instead, our neighborhood buffer (NB) minimizes buffering, uses single-ported memories, and performs one write and read access per cycle while maintaining throughput. As Figure 9 shows, the NB comprises the Transpose Buffer and the neighborhood FIFO. *wAddr* and *rAddr* denote the writing and reading addresses of the transposed buffer, respectively. *byteEn* selects a column in line *wAddr* of the transposed buffer for writing, while all other columns stay unchanged. The incoming data (one value per cycle) is written into the transpose buffer column-wise. The samples from a *row* of channels in the probe are organized as a *column*. The width of the transpose buffer is $2N_r + 1$ the same as the width of a neighborhood, while its depth equals to the width of the probe matrix, P_W. Since our target neural probe has $C = 10K$ channels, P_W may reach 100 for a square-shaped probe. Once the last element of the neighborhood is written to the transpose buffer, the whole neighborhood is in the latest $2N_r + 1$ lines. These lines are read sequentially just ahead enough and pushed into the neighborhood staging FIFO. For example, in Figure 9 the row containing 4–36 is read out from the transpose buffer when 37 is written into it, whereas the line containing 5–37 is read out when writing 38. At that point, the neighborhood staging FIFO contains the neighborhood for 19 which can be whitened. The transpose buffer is implemented as several single-ported SRAM banks. Each cycle, we write a single filtered value to one bank and read a line from another.

Whitening: A channel i's whitened value is the dot-product of all its neighbors and a precomputed whitening matrix, $whitened(i) = neighbors(i) \cdot whiteningMatrix(i)$. The whitening stage receives *en masse* the channel-wise serialized, filtered data from the neighborhood FIFO via dedicated connections, reads the corresponding whitening matrix, and performs a dot-product. The per channel whitening matrices are stored in a C row SRAM, where row i contains the $(2N_r + 1)^2$ whitening coefficients for the neighborhood around channel i. Each neighborhood contains 9 values and the whitening matrices table contains $10K \times 9$ FP32 coefficients. Whitening produces one FP32 value per cycle.

Stages Prior to Template Matching: Before performing template matching we need to: 1) detect that a channel has a spike, 2) determine the central channel - spikes may be picked up by several neighboring electrodes - and 3) send the samples from the neighborhood for template matching. Specifically, once we determine that a spike occurred in channel c and a time t, template matching will need the samples from 9 channels (the central channel and the 8 neighbors surrounding it — e.g., channels 19 and 10-12, 18, 20, and 26-28 respectively in Figure 9). From each of those channels we need 60 samples around t (20 before and 39 after). This is implemented as follows: 1) A spike manifests as a peak which we first detect locally within a channel. As per Figure 2, peaks occur when a sample is larger than ±10 samples in time. This detection is done for *all* channels in "Sample Buffering" and "Peak Detection" stages. This stage also buffers the full window that template matching needs once the central channel is identified. 2) A detected peak is a true peak if none of its neighbors has a higher peak within 10 time steps, which we check for in two stages: 1) First, the "Spike Aging" stage ensures the peak "matures" (stays in the buffer for 10 samples before checking with its neighbors). B) Second, the"Neighborhood Peak" checks if the spike is the highest amongst its neighbors within the ±10 samples.

Sample Buffering and Local Peak Detection: A spike is pivoted by the centered *peak* detected by a sample that exceeds a per channel threshold *and* is greater than ±10 neighboring samples in time (see Figure 2). Once the peak is detected, we pass along the 60 samples around it (the full

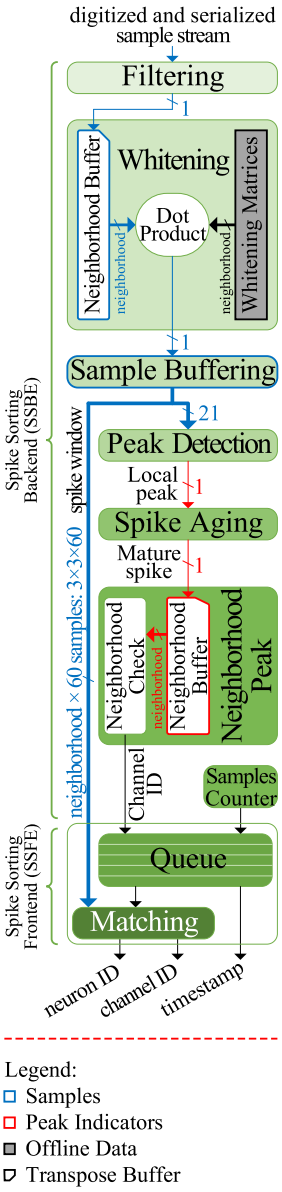

Figure 8. The spike sorting pipeline.

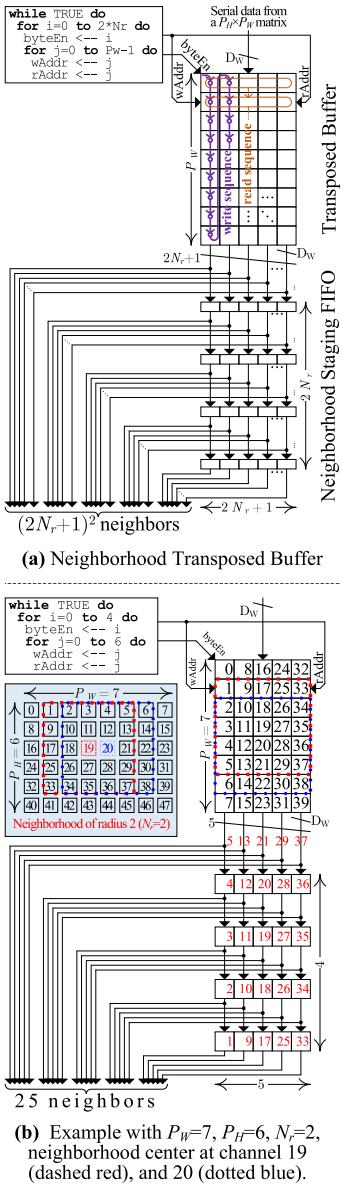

Figure 9. Neighborhood Transpose Buffer.

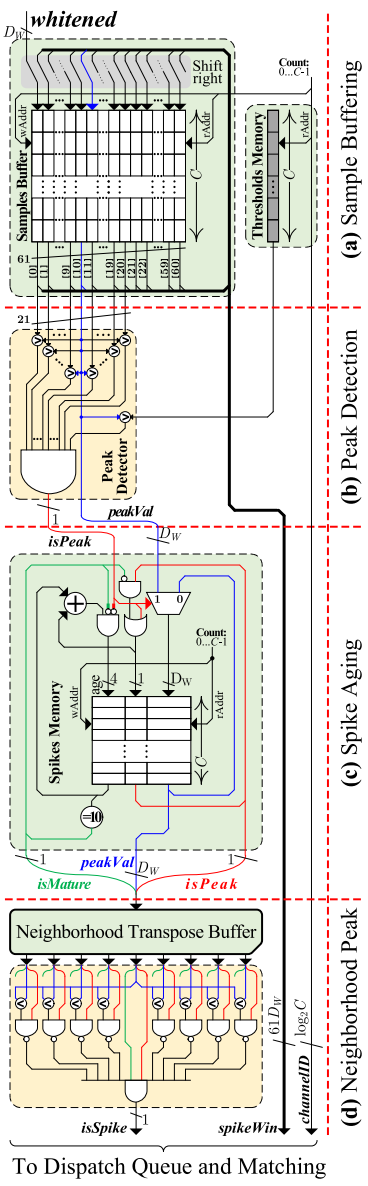

Figure 10. Spike Sorting Stages.

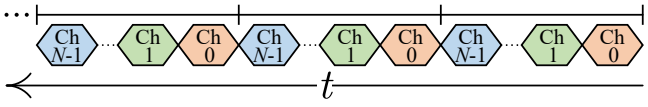

Figure 11. The schedule of the incoming data. Samples from each channel are digitized and serialized. The outputs of the filtering and the whitening stages follow the same schedule.

spike window). We implement this functionality by buffering the last 60 samples per channel in the *Sample Buffer* (SB) shown in Figure 10(a). The SB contains C rows, one per channel. The whitened values are written in the first column of the SB one at a time. In steady state, a full row (60 samples) is read out each cycle, shifted right to include the new incoming whitened sample, and written back to the buffer (a cycle later to allow single ported memories). The *Peak Detector* determines whether a peak has occurred in the 21 most recent samples by comparing the 11th sample (center) with the 10 before and after it and with a per channel threshold. If a peak is detected, the channel number (*ChannelID*), peak indicator (*isPeak*), and the peak value (*peakVal*) proceed to the spike aging counter (SAC) stage which aids with neighborhood peak detection. The SAC ensures that during the next 10 timesteps the magnitude of this local peak

is compared against any other locally detected peaks in the neighboring channels. If this peak happens to be on the central channel, then an entry is pushed in the Matching stages FIFO. At that point, the peak will be in position 20 (as the row has shifted by 10 positions). Template matching occurs 30 timesteps later when the peak will be appropriately centered for peak detection, reading the corresponding samples directly from the SB.

Spike Peak Aging, Maturity, and Expiry: Once a spike is detected, it is necessary to check that none of its neighbors also have spikes within a 10 sample timeframe. To perform this check, the age of each spike is stored and maintained in the *Spikes Memory* as shown in Figure 10(c). Each row in the *Spikes Memory* includes three fields, the relative age of the spike (in samples, 0–10), a single bit indicating a peak, and the peak value. There is one row per channel. If an input peak is detected (*isPeak* is asserted), the age field will be zeroed, the peak indicator will be set high, and the peak value (*peakVal*) in the input will be stored into the memory line corresponding to the same channel. In the subsequent samples of the same channel, the age will be increased until it reaches maturity, namely, 10 sampling cycles. When a peak entry matures, its three fields (maturity indicator *isMature*, *isPeak*, and *peakVal*) are written into the transpose buffer of the neighborhood peak detection stage.

Neighborhood Peak Detection: The purpose of this stage is to check for each spiking channel that no neighboring channel also has spike with a larger magnitude within a 10 samples timeframe. Figure 10(d) shows that this stage is composed of two elements, a transpose buffer accepting entries from the aging unit, and a neighborhood check that performs the neighborhood check. Since the neighborhood is 3×3 the transpose buffer is organized as P_w rows (3 entries each). Each entry contains a peak value, and peak and maturity indicators. Using a similar access strategy to the NB, the 3×3 entries are read into the output FIFO where the check occurs for the entry in the center. If the test succeeds, the spike indicator *isSpike* is asserted, and an entry is placed in the Dispatch queue and tagged with a 40bit counter for identification. Once the full sample window (*spikeWin*) has entered the SB (delaying 10 more timesteps to center the window at the peak), the Matching stage will copy the samples and perform template matching.

Matching: This stage accepts a window of 60 samples per channel from a 3×3 neighborhood of channels (*spikeWin*). The samples are copied from the SB using the *ChannelID* from the dispatch queue. The dispatch queue contains αC entries, where $0 < \alpha \ll 1$ as spikes occur relatively infrequently. For our datasets setting $\alpha = 0.04$ results in no stalls. This stage performs template matching - a dot product of the $3 \times 3 \times 60$ samples from the Samples Buffer with one or more templates. The center channel index is used to fetch the templates. The matching neuron corresponds to the highest magnitude dot product. We implement this unit as a vector datapath comprising several multiply accumulate units. A 16-wide datapath ensures that the matching stage can process incoming spikes at the exceedingly rare peak rate of 20Hz per neuron and 13 templates per channel.

Template Decompression: The templates are stored as a fixed and variable portion, decompressing on demand using the *ChannelID*. Recall that every template contains 60 samples per channel (hitherto referred to as a *waveform*) across 9 channels. The fixed storage consists of a 4b centroid tag, and six 9b metadata chunks (5b bases + 1b DF + 3b lengths) - for a total of 58b per waveform. A row of fixed memory is then stored as nine 58b segments (by 30,000 columns, the number of templates) accessed with 15b indices. The number of templates for that channel can be inferred as the difference between the current and next index, which is commonly 3 but can be as much as 13. The variable storage consists of the variable length Δ. To store these, we use a 9×9 grid of memory blocks ($\Delta \times$neighbors). Δ are packed in 9 virtual columns in segment order allowing efficient expansion into 5b [15]. Having 9×9 memory blocks enables parallel access to each of the 9 values of a segment, and each of the 9 waveforms. Since the 9 values of a segment have the same bitwidth, we can load all segments of a template in 6 cycles.

To fully decompress a single waveform, the 4b centroid tag extracts a centroid waveform, i.e. sixty 12b values from the 16-row centroid table. In parallel, the segments can be loaded as above. The base is forwarded to a Δ decoder. For DF= 0, the length corresponds to the size of each of nine Δ ([0,5b]). For DF= 1, the Δ are treated as a 9b pointer to a 512-entry table with nine 2b Δ which are the segment. Each Δ is consecutively added to the base to reproduce the index to the 32-entry quantization codebook. The *codebook value* is finally added to the corresponding *centroid value* to reproduce the original *template value*. Parallel access is used for the Δ decoder and codebook for acceleration. Outliers act as an override for the decompressed value from the codebook, as outliers must still be added to the centroid. A maximum of 1.1% of all values (178k) were classified as outliers. We pessimistically provision for up to 200k outliers. When loading templates, we locate the number of waveforms that contain outliers. This is inferred by reading two consecutive entries of an outlier pointer memory for the starting count and the subsequent count of waveforms with outliers for the current template, and indexes into the offsets needed to locate the position and outlier value. Since only 10% of waveforms have outliers, the index only needs 27k entries of 4b for the segment and 19b for the offset into the larger 200k memories. The outlier position is 6b (for its position in the template), and the value itself is 7b.

5.2 Architectural Evaluation

To evaluate Marple, a configuration with 10K channels was implemented using a commercial 65nm process. This configuration is capable of supporting low latency BMIs, which

Table 5. Design Parameters and Memory Blocks

Parameter	Value	Description
Q	12 bits (typical)	ADC resolution
C	10,000 (target)	Number of channels
N	30,000 (target)	Number of neurons
f_s	30KHz (typical)	Sampling rate
$f_{op} = Nf_s$	300MHz (typical)	Operating frequency
P_W	100 (square probe)	Neural probe width (channels)
P_H	100 (square probe)	Neural probe height (channels)
N_r	1 (typical)	Neighborhood radius
N_b	9 (typical)	Neighborhood size
α	0.04 (modeled)	Dispatch queue size factor
D_W	32	Data width

Stage/Unit	Size		
	Parametric	Target	MBits
Filtering			
Filter Scratchpad	$C \times 6D_W$	10000×192	1.831
Whitening			
Transpose Buffer	$P_W \times (2N_r+1)D_W$	100×96	0.009
Whitening Matrix	$C \times (2N_r+1)^2 D_W$	10000×288	2.747
Sample Buffering and Peak Detection			
Samples Buffer	$C \times 61D_W$	10000×1952	18.61
Thresholds Memory	$C \times D_W$	10000×32	0.305
Spike Aging			
Spikes Memory	$C \times (5+D_W)$	10000×37	0.353
Neighborhood Merge			
Transpose Buffer	$P_W \times (2N_r+1)(D_W+2)$	100×102	0.001
Dispatch Queue			
FIFO Memory	$\alpha C \times (\log_2 C + 40)$	400×54	0.0206
Template Matching + Decompression			
Fixed Count	$C \times \log_2 N$	10000×15	0.143
Variable Offset	$C \times \log_2 N$	10000×22	0.210
Fixed Memory	$N \times N_b \times (4+9\times 6)$	30000×522	14.93
Variable Memory	$N \times N_b \times 5 \times 4 \times 9$	30000×1620	46.35
Centroids Memory	$\log_2 C \times S \times N_b \times 12$	$16 \times 720 \times 9$	0.099
Duplicates Memory	-	$512 \times 18 \times 9$	0.079
Quantization Codebook	-	$32 \times 32 \times 54$	0.053
Outlier Pointer Memory	-	30000×15	0.429
Outlier Count	-	30000×15	0.429
Outlier Index	-	27000×23	0.592
Outlier Positions	-	200000×6	1.14
Outlier Values	-	200000×7	1.34
Total	-	-	89.37

require a sampling rate of 30 kHz. The target operation frequency is set to 300 MHz for achieve optimal performance. We implement the units in Verilog and synthesize with the Synopsys Design Compiler. *Layout* uses Cadence Encounter and Synopsys' commercial *Building Block IP* library. We estimate power via Encounter. We use nominal operating conditions to model power and latency. We model SRAM buffers using CACTI [46]. Table 6 summarizes the post-layout logic and memory costs for each of the modules to quantify the area and power consumption. Both area and power are dominated by the memories in *Sample Buffering + Peak Detection* and *Template Matching + Decompression*, accounting for 18% & 77% of total area and 44% & 33% of power, respectively. However, much of the power costs are due to standby leakage (598mW, 45%). We estimate Marple's power use and area with more recent technologies using the methodology of Stillmaker and Baas [80]. Table 7 shows total power and area estimates with technology nodes varying up to 7nm. Scaling Marple to 7nm would reduce the area and power to 4.25mm^2 and 78.94mW, respectively. Due to the specificity and constraints of a portable online spike sorter, Marple requires fine-grained customization for accurate implementation.

Neural network-based spike sorting: We explore an alternative to template matching by using a Convolutional Neural Network (CNN). The CNN accepts the same input as template matching, and the *ChannelID*, outputting a vector for the firing neurons ($\mathbb{R}^{neurons}$). Table 8 details the model's architecture (where applicable the stride is 2), 3 configurations evaluated, and the compute and memory costs during inference. Hyperparameters for model size and training were empirically derived. Training times range from 2-12 hours on a NVIDIA GeForce RTX 3090 GPU. Performance is measured on the NP datasets as the 5-fold cross validation accuracy. All models outperform template matching: Template matching's accuracy for these extremely large datasets is 67% whereas the small, middle, and large CNNs achieve accuracies of 85.6%, 89.9% and 91.9%, respectively. However, practical deployment of CNNs are difficult - memory demands of even the *small* models exceed template matching for 30K neurons. Worse, Figure 12 shows the minimum computation bandwidth that is needed for the *small* model for different firing rates and neuron counts. The 1K neuron configuration with the *small* model could be practical for simple applications as it requires 1.48GOPs and about 1.6MB of storage. However, with 30K neurons (10K channels) the demands exceed 1.48TOPs even with the lower $F = 5$.

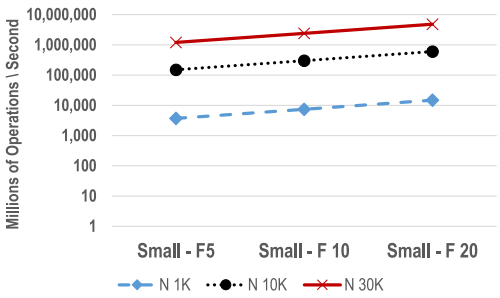

Figure 12. "Small" CNN: Computation demand scaling with the number of neurons N and firing rate F.

Many recent works use neural networks for spike sorting [16, 38, 39, 60, 68, 69], however, none evaluate performance on the scale at which we do (the largest is 128 channels [39], two orders of magnitude less). While more accurate, CNNs are only appropriate for very small configurations or offline applications. Our analysis serves as motivation for further work to refine the CNN-based approach.

6 Related Work

Prior work can be divided into software and hardware solutions. Software solutions are the *state-of-the-art*: they provide high accuracy, performing well for large channel counts, and

Table 6. Post place-and-route area & power estimates for Marple (@ 65nm)

Stage	Area (Logic)	Area (Memory)	Area (Total)	Power (Logic)	Power (Memory)	Power (Total)
	mm^2	mm^2	mm^2	mW	mW	mW
Filtering	0.122	1.532	1.654	35.22	46.36	81.58
Whitening (Pipelined)	0.225	2.926	3.151	62.72	69.04	131.76
Sample Buffering + Peak Detection	0.034	19.436	19.471	7.48	576.25	583.73
Spike Aging + Neighborhood Peak	0.005	0.366	0.371	0.63	5.51	6.14
Dispatch Queue	0.028	0.04	0.068	12.86	0.23	13.09
Template Matching + Decompression	0.447	83.869	84.316	171.00	437.21	608.21
Total	**0.861**	**108.169**	**109.03**	**289.9**	**1134.6**	**1424.51**

Table 7. Scaling technology nodes

Tech. Node	Power (mW)	Total Area (mm^2)
65nm	1424.51	109.03
45nm	881.28	71.96
32nm	443.93	33.8
20nm	256.02	15.26
16nm	172.38	14.17
14nm	133.39	13.08
10nm	107.1	7.41
7nm	78.94	4.25

Table 8. CNNs configuration, architecture, compute & memory costs, and accuracy.

Model Configurations

Parameter	*Small*	*Medium*	*Large*
$n/i/j/k$	16 / 513 / 256 / 128	32 / 1025 / 512 / 256	64 / 2049 / 1024 / 512

Model Architecture

Layer	Type	Dimensions
1	1D Conv	$n \times 58$
2	Max Pool	$n \times 29$
3	Squeeze Exc.	$n \times 29$
4	1D Conv	$2 \times n \times 29$
5	Max Pool	$2 \times n \times 14$
6	Adapt. Avg. Pool	$1 \times i + channelID$
7	Fully Conn.	$1 \times j$
8	Fully Conn.	$1 \times k$
9	Fully Conn.	$1 \times N$

Compute and Memory Costs to Accuracy

Model	Neurons	FLOPs	Elements (FP32)	Accuracy
Small	1K	0.74M	0.37M	93.5%
	10K	3.04M	1.54M	88.0%
	30K	8.03M	4.08M	85.6%
Medium	1K	2.13M	1.07M	94.1%
	10K	6.74M	3.39M	91.6%
	30K	16.72M	8.43M	89.9%
Large	1K	6.88M	3.47M	94.5%
	10K	16.10M	8.10M	92.3%
	30K	36.06M	18.12M	91.9%

are widely used in post-hoc analysis [10, 26, 54, 94]. However, they are running primarily on desktop GPUs or server class hardware, incurring large energy costs and reduced portability. Additionally, most of them cannot operate in real-time for large channel counts. Existing hardware solutions [71, 87, 93] sacrifice accuracy and scalability for the sake of implementation and form factors. The closest hardware design to Marple is from Valencia and Alimohammad (VA) [87]. Compared to Marple, their spike sorter uses the NEO spike detector [31, 93], and performs template matching with OSort [65] only on *single-channels*. The VA design favours hardware simplicity forgoing more accurate methods such as whitening and employing spatial neighborhoods [10, 53, 54]. However, given more than a decade since OSort's conception, this sacrifices scalability and accuracy. For high-density probes, this is problematic since: 1) neurons are often detected on multiple probes (one neuron is picked up by many probes), 2) having several probes in close proximity allows us to discern among multiple neuron groups that are nearby (many neurons are picked up by several probes in a way that allows us to discern which one it was). Therefore, the VA design is inappropriate for such setups because: 1) it will detect each spike multiple times, once per neighboring channel. 2) It will be unable to discern among multiple neurons that are detected from the same electrode. VA is implemented in 45nm and requires half the power but 30x more area than Marple when scaled up to 10K channels, as they focus on single-channel analysis. Schäffer et al. [71] incorporate multiple channels to mitigate this problem, but still utilize NEO and OSort as the baseline algorithms, and incur similar shortcomings to VA. Their implementation scales poorly with channels, as it performs global comparisons with *every other channel* in the system for template matching, rather than locally. Other hardware solutions [93] perform a subset of stages such as spike detection but not spike sorting, thus targeting different types of applications [30]. Overall, prior hardware solutions handle only input from *few* channels, limiting real-world applications where *coarse-grain* neural decoding is sufficient and, therefore, allows for use of only very simplistic spike sorting methods [7]. Instead, Marple i) effectively handles a large number of channel inputs, ii) affords using advanced spike sorting methods, and iii) covers a wide variety of real-worlds applications, thus significantly assisting to increase the development of impactful BMIs.

7 Conclusion

Scalability is a pressing problem with modern neural recording devices, requiring novel software and hardware solutions to keep pace [7, 35, 63]. We analyzed the computational and memory bottlenecks for untethered, real-time spike sorting and concluded that commodity platforms are not suitable for wearable applications. We developed: 1) a novel, lightweight purpose-built template compression method, and 2) an accelerator for performing the computations in real-time. We further explored using CNNs to improve spike sorting accuracy on large-scale recordings. Since Marple is modularly designed, its constituents can be independently optimized for specific constraints. Other systems may find it useful to integrate parts of the design for low-power, end-to-end solutions [30], and solutions for alternate recording types such as EEG [92].

References

[1] William E. Allen, Michael Z. Chen, Nandini Pichamoorthy, Rebecca H. Tien, Marius Pachitariu, Liqun Luo, and Karl Deisseroth. 2019. Thirst regulates motivated behavior through modulation of brainwide neural population dynamics. *Science* 364, 6437 (2019), eaav3932. https://doi.org/10.1126/science.aav3932

[2] Yehia Arafa, Ammar ElWazir, Abdelrahman Elkanishy, Youssef Aly, Ayatelrahman Elsayed, Abdel-Hameed Badawy, Gopinath Chennupati, Stephan Eidenbenz, and Nandakishore Santhi. 2020. Verified instruction-level energy consumption measurement for NVIDIA GPUs. *Proceedings of the 17th ACM International Conference on computing frontiers* (2020), 60–70. https://doi.org/10.1145/3387902.3392613

[3] Bruno B. Averbeck, David A. Crowe, Matthew V. Chafee, and Apostolos P. Georgopoulos. 2003. Neural activity in prefrontal cortex during copying geometrical shapes. II. Decoding shape segments from neural ensembles. *Experimental Brain Research* 150, 2 (2003), 142–153. https://doi.org/10.1007/s00221-003-1417-5

[4] Réka Barbara Bod, János Rokai, Domokos Meszéna, Richárd Fiáth, István Ulbert, and Gergely Márton. 2022. From End to End: Gaining, Sorting, and Employing High-Density Neural Single Unit Recordings. *Frontiers in Neuroinformatics* 16 (2022). https://doi.org/10.3389/fninf.2022.851024

[5] Alessio Paolo Buccino and Gaute Tomas Einevoll. 2020. Mearec: a fast and customizable testbench simulator for ground-truth extracellular spiking activity. *Neuroinformatics* (2020), 1–20.

[6] Alessio Paolo Buccino, Samuel Garcia, and Pierre Yger. 2022. Spike sorting: new trends and challenges of the era of high-density probes. *Progress in Biomedical Engineering* 4, 2 (5 2022), 022005. https://doi.org/10.1088/2516-1091/ac6b96

[7] David Carlson and Lawrence Carin. 2019. Continuing progress of spike sorting in the era of big data. *Current Opinion in Neurobiology* 55 (2019), 90–96. https://doi.org/10.1016/j.conb.2019.02.007 Machine Learning, Big Data, and Neuroscience.

[8] Nicholas T. Carnevale and Michael L. Hines. 2006. *The NEURON Book*. Cambridge University Press. https://doi.org/10.1017/CBO9780511541612

[9] Fernando J. Chaure, Hernan G. Rey, and Rodrigo Quian Quiroga. 2018. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of Neurophysiology* 120, 4 (2018), 1859–1871. https://doi.org/10.1152/jn.00339.2018 PMID: 29995603.

[10] Jason E. Chung, Jeremy F. Magland, Alex H. Barnett, Vanessa M. Tolosa, Angela C. Tooker, Kye Y. Lee, Kedar G. Shah, Sarah H. Felix, Loren M. Frank, and Leslie F. Greengard. 2017. A Fully Automated Approach to Spike Sorting. *Neuron* 95, 6 (2017), 1381–1394.e6. https://doi.org/10.1016/j.neuron.2017.08.030

[11] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Stephen I Ryu, and Krishna V Shenoy. 2012. Neural population dynamics during reaching. *Nature* 487, 7405 (2012), 51–56. https://doi.org/10.1038/nature11129

[12] Davide Ciliberti, Frédéric Michon, and Fabian Kloosterman. 2018. Real-time classification of experience-related ensemble spiking patterns for closed-loop applications. *eLife* 7 (10 2018), e36275. https://doi.org/10.7554/eLife.36275

[13] NVIDIA Corporation. [n. d.]. Nvidia Jetson Orin. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

[14] Elodie Despouy, Jonathan Curot, Leila Reddy, Lionel G. Nowak, Martin Deudon, Jean-Christophe Sol, Jean-Albert Lotterie, Marie Denuelle, Ali Maziz, Christian Bergaud, Simon J. Thorpe, Luc Valton, and Emmanuel J. Barbeau. 2020. Recording local field potential and neuronal activity with tetrodes in epileptic patients. *Journal of Neuroscience Methods* 341 (2020), 108759. https://doi.org/10.1016/j.jneumeth.2020.108759

[15] Isak Edo Vivancos, Sayeh Sharify, Daniel Ly-Ma, Ameer Abdelhadi, Ciaran Bannon, Milos Nikolic, Mostafa Mahmoud, Alberto Delmas Lascorz, Gennady Pekhimenko, Andreas Moshovos, and et al. 2021. Boveda: Building an On-Chip Deep Learning Memory Hierarchy Brick by Brick. *Proceedings of Machine Learning and Systems* (3 2021). https://proceedings.mlsys.org/paper/2021/hash/013d407166ec4fa56eb1e1f8cbe183b9-Abstract.html

[16] Junsik Eom, In Yong Park, Sewon Kim, Hanbyol Jang, Sanggeon Park, Yeowool Huh, and Dosik Hwang. 2021. Deep-learned spike representations and sorting via an ensemble of auto-encoders. *Neural Networks* 134 (2021), 131–142. https://doi.org/10.1016/j.neunet.2020.11.009

[17] Nir Even-Chen, Dante G Muratore, Sergey D Stavisky, Leigh R Hochberg, Jaimie M Henderson, Boris Murmann, and Krishna V Shenoy. 2020. Power-saving design opportunities for wireless intracortical brain-computer interfaces. *Nature biomedical engineering* 4, 10 (10 2020), 984—996. https://doi.org/10.1038/s41551-020-0595-9

[18] Richárd Fiáth, Bogdan Cristian Raducanu, Silke Musa, Alexandru Andrei, Carolina Mora Lopez, Chris van Hoof, Patrick Ruther, Arno Aarts, Domonkos Horváth, and István Ulbert. 2018. A silicon-based neural probe with densely-packed low-impedance titanium nitride microelectrodes for ultrahigh-resolution in vivo recordings. *Biosensors and Bioelectronics* 106 (2018), 86–92. https://doi.org/10.1016/j.bios.2018.01.060

[19] Apostolos P Georgopoulos, Andrew B Schwartz, Ronald E Kettner, C T R Wilson, and J Franklin Inst. 1986. Neuronal Population Coding of Movement Direction. *American Association for the Advancement of Science* 233, 4771 (1986), 1416–1419.

[20] Felipe Gerhard, Tilman Kispersky, Gabrielle J. Gutierrez, Eve Marder, Mark Kramer, and Uri Eden. 2013. Successful Reconstruction of a Physiological Circuit with Known Connectivity from Spiking Activity Alone. *PLoS Computational Biology* 9, 7 (2013), 32–34. https://doi.org/10.1371/journal.pcbi.1003138

[21] Michael Hines, Andrew Davison, and Eilif Muller. 2009. NEURON and Python. *Frontiers in Neuroinformatics* 3 (2009). https://doi.org/10.3389/neuro.11.001.2009

[22] Leigh R. Hochberg, Mijail D. Serruya, Gerhard M. Friehs, Jon A. Mukand, Maryam Saleh, Abraham H. Caplan, Almut Branner, David Chen, Richard D. Penn, and John P. Donoghue. 2006. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature* 442 (2006), 164–171. https://doi.org/10.1038/nature04970

[23] Intel. [n. d.]. Core i9-9900K Processor specifications. https://ark.intel.com/content/www/us/en/ark/products/186605/intel-core-i99900k-processor-16m-cache-up-to-5-00-ghz.html

[24] ISO 14708-3:2017 2017. *Implants for surgery: Active implantable medical devices. Part 3: Implantable neurostimulators.* Standard. International Organization for Standardization, Geneva, CH.

[25] Ferris Jabr. 2022. The Man Who Controls Computers With His Mind. *The New York Times* (2022). https://www.nytimes.com/2022/05/12/magazine/brain-computer-interface.html

[26] James J. Jun, Catalin Mitelut, Chongxi Lai, Sergey L. Gratiy, Costas A. Anastassiou, and Timothy D. Harris. 2017. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv* (2017), 1–33. https://doi.org/10.1101/101030

[27] James J. Jun, Nicholas A. Steinmetz, Joshua H. Siegle, Daniel J. Denman, Marius Bauza, Brian Barbarits, Albert K. Lee, Costas A. Anastassiou, Alexandru Andrei, Çağatay Aydin, Mladen Barbic, Timothy J. Blanche, Vincent Bonin, João Couto, Barundeb Dutta, Sergey L. Gratiy, Diego A. Gutnisky, Michael Häusser, Bill Karsh, Peter Ledochowitsch, Carolina Mora Lopez, Catalin Mitelut, Silke Musa, Michael Okun, Marius Pachitariu, Jan Putzeys, P. Dylan Rich, Cyrille Rossant, Wei Lung Sun, Karel Svoboda, Matteo Carandini, Kenneth D. Harris, Christof Koch, John O'Keefe, and Timothy D. Harris. 2017. Fully integrated silicon probes for high-density recording of neural activity. *Nature* 551, 7679 (2017), 232–236. https://doi.org/10.1038/nature24636

[28] Florek K., Łukaszewicz, J. Steinhaus Hugo J. Perkal, and Zubrzycki S. 1951. Sur la liaison et la division des points d'un ensemble fini.

Colloquium Mathematicum 2, 3-4 (1951), 282–285. http://eudml.org/doc/209969

[29] Heet Kaku, Musa Ozturk, Ashwin Viswanathan, Joohi Jimenez-Shahed, Sameer Sheth, and Nuri F Ince. 2019. Grouping Neuronal Spiking Patterns in the Subthalamic Nucleus of Parkinsonian Patients. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 4221–4224. https://doi.org/10.1109/EMBC.2019.8857418

[30] Ioannis Karageorgos, Karthik Sriram, Ján Veselý, Michael Wu, Marc Powell, David Borton, Rajit Manohar, and Abhishek Bhattacharjee. 2020. Hardware-Software Co-Design for Brain-Computer Interfaces. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 391–404. https://doi.org/10.1109/ISCA45697.2020.00041

[31] Vaibhav Karkare, Sarah Gibson, and Dejan Marković. 2011. A 130-μ W, 64-Channel Neural Spike-Sorting DSP Chip. *IEEE Journal of Solid-State Circuits* 46, 5 (2011), 1214–1222. https://doi.org/10.1109/JSSC.2011.2116410

[32] Thong-Wei Koh, Paul A. Merolla, Sonal Pinto, and Dongjin Seo. 2023. Real-time Neural Spike Detection. https://patents.google.com/patent/CA3146297C/en Patent CA3146297C.

[33] Andrea A. Kühn, Thomas Trottenberg, Anatol Kivi, Andreas Kupsch, Gerd-Helge Schneider, and Peter Brown. 2005. The relationship between local field potential and neuronal discharge in the subthalamic nucleus of patients with Parkinson's disease. *Experimental Neurology* 194, 1 (2005), 212–220. https://doi.org/10.1016/j.expneurol.2005.02.010

[34] Mikhail A Lebedev and Miguel A L Nicolelis. 2006. Brain–machine interfaces: past, present and future. *Trends Neurosci.* 29, 9 (2006), 536–546.

[35] Baptiste Lefebvre, Pierre Yger, and Olivier Marre. 2016. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology Paris* 110 (11 2016), 327–335. Issue 4. https://doi.org/10.1016/j.jphysparis.2017.02.005

[36] Michael S. Lewicki. 1998. A review of methods for spike sorting: The detection and classification of neural action potentials. *Network: Computation in Neural Systems* 9, 4 (1998). https://doi.org/10.1088/0954-898X_9_4_001

[37] Hsin-Hung Li and Clayton E. Curtis. 2023. Neural population dynamics of human working memory. *Current Biology* 33, 17 (2023), 3775–3784.e4. https://doi.org/10.1016/j.cub.2023.07.067

[38] Jie Li, Xiuli Chen, and Zheng Li. 2018. Spike detection and spike sorting with a hidden Markov model improves offline decoding of motor cortical recordings. *Journal of Neural Engineering* 16, 1 (12 2018), 016014. https://doi.org/10.1088/1741-2552/aaeaae

[39] Zhaohui Li, Yongtian Wang, Nan Zhang, and Xiaoli Li. 2020. An Accurate and Robust Method for Spike Sorting Based on Convolutional Neural Networks. *Brain Sciences* 10, 11 (2020). https://doi.org/10.3390/brainsci10110835

[40] Yu Po Lin, Chun Yi Yeh, Pin Yang Huang, Zong Ye Wang, Hsiang Hui Cheng, Yi Ting Li, Chi Fen Chuang, Po Chiun Huang, Kea Tiong Tang, Hsi Pin Ma, Yen Chung Chang, Shih Rung Yeh, and Hsin Chen. 2016. A Battery-Less, Implantable Neuro-Electronic Interface for Studying the Mechanisms of Deep Brain Stimulation in Rat Models. *IEEE Transactions on Biomedical Circuits and Systems* 10 (2 2016), 98–112. Issue 1. https://doi.org/10.1109/TBCAS.2015.2403282

[41] Jeremy F. Magland, James J. Jun, Elizabeth Lovero, Alexander J. Morley, Cole L. Hurwitz, Alessio P. Buccino, Samuel Garcia, and Alex H. Barnett. 2020. SpikeForest: Reproducible web-facing ground-truth validation of automated neural spike sorters. *bioRxiv* i (2020), 1–22. https://doi.org/10.1101/2020.01.14.900688

[42] Edwin M. Maynard, Craig T. Nordhausen, and Richard A. Normann. 1997. The Utah Intracortical Electrode Array: A recording structure for potential brain-computer interfaces. *Electroencephalography and Clinical Neurophysiology* 102, 3 (1997), 228–239. https://doi.org/10.1016/S0013-4694(96)95176-0

[43] Louis L. Mcquitty. 1957. Elementary Linkage Analysis for Isolating Orthogonal and Oblique Types and Typal Relevancies. *Educational and Psychological Measurement* 17 (1957), 207 – 229.

[44] Edward M. Merricks, Elliot H. Smith, Guy M. McKhann, Robert R. Goodman, Lisa M. Bateman, Ronald G. Emerson, Catherine A. Schevon, and Andrew J. Trevelyan. 2015. Single unit action potentials in humans and the effect of seizure activity. *Brain* 138, 10 (2015), 2891–2906. https://doi.org/10.1093/brain/awv208

[45] Blake A. Mitchell, Brock M. Carlson, Jacob A. Westerberg, Michele A. Cox, and Alexander Maier. 2023. A role for ocular dominance in binocular integration. *Current Biology* 33, 18 (2023), 3884–3895.e5. https://doi.org/10.1016/j.cub.2023.08.019

[46] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman Jouppi. 2009. Cacti 6.0: A tool to model large caches. *HP Laboratories* (01 2009).

[47] Elon Musk and Neuralink. 2019. An integrated brain-machine interface platform with thousands of channels. *bioRxiv* (2019). https://doi.org/10.1101/703801

[48] Joaquin Navajas, Deren Y. Barsakcioglu, Amir Eftekhar, Andrew Jackson, Timothy G. Constandinou, and Rodrigo Quian Quiroga. 2014. Minimum requirements for accurate and efficient real-time on-chip spike sorting. *Journal of Neuroscience Methods* 230 (2014), 51–64. https://doi.org/10.1016/j.jneumeth.2014.04.018

[49] Joana P. Neto, Gonçalo Lopes, João Frazão, Joana Nogueira, Pedro Lacerda, Pedro Baião, Arno Aarts, Alexandru Andrei, Silke Musa, Elvira Fortunato, Pedro Barquinha, and Adam R. Kampff. 2016. Validating silicon polytrodes with paired juxtacellular recordings: Method and dataset. *Journal of Neurophysiology* 116, 2 (2016), 892–903. https://doi.org/10.1152/jn.00103.2016

[50] NeuroNexus. 2021. NeuroNexus 2021 Probe catalog. Catalog. https://www.neuronexus.com/files/catalog/2021-Probe-Catalog.pdf

[51] Arto Nurmikko. 2020. Challenges for Large-Scale Cortical Interfaces. *Neuron* 108, 2 (2020), 259–269. https://doi.org/10.1016/j.neuron.2020.10.015

[52] Abdulmalik Obaid, Mina-Elraheb Hanna, Yu-Wei Wu, Mihaly Kollo, Romeo Racz, Matthew R Angle, Jan Müller, Nora Brackbill, William Wray, Felix Franke, E J Chichilnisky, Andreas Hierlemann, Jun B Ding, Andreas T Schaefer, and Nicholas A Melosh. 2020. Massively parallel microwire arrays integrated with CMOS chips for neural recording.

[53] Marius Pachitariu, Shashwat Sridhar, and Carsen Stringer. 2023. Solving the spike sorting problem with Kilosort. *bioRxiv* (2023). https://doi.org/10.1101/2023.01.07.523036

[54] Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, and Harris Kenneth D. 2016. Realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *bioRxiv* (2016), 061481. https://doi.org/10.1101/061481

[55] Stefano Panzeri, Jakob H. Macke, Joachim Gross, and Christoph Kayser. 2015. Neural population coding: combining insights from microscopic and mass signals. *Trends in Cognitive Sciences* 19, 3 (2015), 162–172. https://doi.org/10.1016/j.tics.2015.01.002

[56] In Yong Park, Junsik Eom, Hanbyol Jang, Sewon Kim, Sanggeon Park, Yeowool Huh, and Dosik Hwang. 2020. Deep learning-based template matching spike classification for extracellular recordings. *Applied Sciences (Switzerland)* 10 (1 2020). Issue 1. https://doi.org/10.3390/app10010301

[57] Gennady Pekhimenko, Vivek Seshadri, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry. 2012. Base-delta-immediate compression: Practical data compression for on-chip caches. *Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT* (2012), 377–388. https://doi.org/10.1145/2370816.2370870

[58] Jan Putzeys, Bogdan C. Raducanu, Alain Carton, Jef De Ceulaer, Bill Karsh, Joshua H. Siegle, Nick Van Helleputte, Timothy D. Harris, Barundeb Dutta, Silke Musa, and Carolina Mora Lopez. 2019.

Neuropixels Data-Acquisition System: A Scalable Platform for Parallel Recording of 10 000+ Electrophysiological Signals. *IEEE transactions on biomedical circuits and systems* 13, 6 (2019), 1635–1644. https://doi.org/10.1109/TBCAS.2019.2943077

[59] Rodrigo Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. 2004. Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering. *Neural Computation* 16, 8 (08 2004), 1661–1687. https://doi.org/10.1162/089976604774201631 arXiv:https://direct.mit.edu/neco/article-pdf/16/8/1661/816020/089976604774201631.pdf

[60] Melinda Rácz, Csaba Liber, Erik Németh, Richárd Fiáth, János Rokai, István Harmati, István Ulbert, and Gergely Márton. 2020. Spike detection and sorting with deep learning. *Journal of Neural Engineering* 17, 1 (1 2020), 016038. https://doi.org/10.1088/1741-2552/ab4896

[61] Adrien B. Rapeaux and Timothy G. Constandinou. 2021. Implantable brain machine interfaces: first-in-human studies, technology challenges and trends. *Current opinion in biotechnology* 72 (12 2021), 102—111. https://doi.org/10.1016/j.copbio.2021.10.001

[62] Thomas P. Reber, Marcel Bausch, Sina Mackay, Jan Boström, Christian E. Elger, and Florian Mormann. 2019. Representation of abstract semantic knowledge in populations of human single neurons in the medial temporal lobe. *PLOS Biology* 17, 6 (06 2019), 1–17. https://doi.org/10.1371/journal.pbio.3000290

[63] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. 2015. Past, present and future of spike sorting techniques. *Brain Research Bulletin* 119 (2015), 106–117. https://doi.org/10.1016/j.brainresbull.2015.04.007

[64] Cyrille Rossant, Shabnam N. Kadir, Dan F.M. Goodman, John Schulman, Maximilian L.D. Hunter, Aman B. Saleem, Andres Grosmark, Mariano Belluscio, George H. Denfield, Alexander S. Ecker, Andreas S. Tolias, Samuel Solomon, György Buzski, Matteo Carandini, and Kenneth D. Harris. 2016. Spike sorting for large, dense electrode arrays. *Nature Neuroscience* 19, 4 (2016), 634–641. https://doi.org/10.1038/nn.4268

[65] Ueli Rutishauser, Erin M. Schuman, and Adam N. Mamelak. 2006. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of Neuroscience Methods* 154, 1 (2006), 204–224. https://doi.org/10.1016/j.jneumeth.2005.12.033

[66] Simanto Saha, Khondaker A. Mamun, Khawza Ahmed, Raqibul Mostafa, Ganesh R. Naik, Sam Darvishi, Ahsan H. Khandoker, and Mathias Baumert. 2021. Progress in Brain Computer Interface: Challenges and Opportunities. *Frontiers in Systems Neuroscience* 15 (2021). https://doi.org/10.3389/fnsys.2021.578875

[67] Kunal Sahasrabuddhe, Aamir A. Khan, Aditya P. Singh, Tyler M. Stern, Yeena Ng, Aleksandar Tadić, Peter Orel, Chris LaReau, Daniel Pouzzner, Kurtis Nishimura, Kevin M. Boergens, Sashank Shivakumar, Matthew S. Hopper, Bryan Kerr, Mina-Elraheb S. Hanna, Robert J. Edgington, Ingrid McNamara, Devin Fell, Peng Gao, Amir Babaie-Fishani, Sampsa Veijalainen, Alexander V. Klekachev, Alison M. Stuckey, Bert Luyssaert, Takashi D. Y. Kozai, Chong Xie, Vikash Gilja, Bart Dierickx, Yifan Kong, Malgorzata Straka, Harbaljit S. Sohal, and Matthew R. Angle. 2020. The Argo: A 65,536 channel recording system for high density neural recording in vivo. *bioRxiv* (2020). https://doi.org/10.1101/2020.07.17.209403

[68] Muhammad Saif-Ur-Rehman, Omair Ali, Susanne Dyck, Robin Lienkämper, Marita Metzler, Yaroslav Parpaley, Jörg Wellmer, Charles Liu, Brian Lee, Spencer Kellis, Richard Andersen, Ioannis Iossifidis, Tobias Glasmachers, and Christian Klaes. 2021. SpikeDeep-classifier: A deep-learning based fully automatic offline spike sorting algorithm. *Journal of Neural Engineering* 18, 1 (2021). https://doi.org/10.1088/1741-2552/abc8d4 arXiv:1912.10749

[69] Muhammad Saif-Ur-Rehman, Robin Lienk mper, Yaroslav Parpaley, Jörg Wellmer, Charles Liu, Brian Lee, Spencer Kellis, Richard Andersen, Ioannis Iossifidis, Tobias Glasmachers, and Christian Klaes. 2019. (SpikeDeeptector) A deep-learning based method for detection of neural spiking activity. *Journal of Neural Engineering* 16, 5 (2019). https://doi.org/10.1088/1741-2552/ab1e63

[70] Shreya Saxena and John P. Cunningham. 2019. Towards the neural population doctrine. *Current Opinion in Neurobiology* 55 (2019), 103–111. https://doi.org/10.1016/j.conb.2019.02.002 Machine Learning, Big Data, and Neuroscience.

[71] Laszlo Schaffer, Zoltan Nagy, Zoltan Kincses, Richard Fiath, and Istvan Ulbert. 2021. Spatial Information Based OSort for Real-Time Spike Sorting Using FPGA. *IEEE Transactions on Biomedical Engineering* 68, 1 (2021), 99–108. https://doi.org/10.1109/TBME.2020.2996281

[72] Changyu Seong, Wonjae Lee, and Dongsuk Jeon. 2021. A Multi-Channel Spike Sorting Processor with Accurate Clustering Algorithm Using Convolutional Autoencoder. *IEEE Transactions on Biomedical Circuits and Systems* 15 (2021), 1441–1453. Issue 6. https://doi.org/10.1109/TBCAS.2021.3134660

[73] Claudia Serrano-Amenos, Payam Heydari, Charles Y Liu, An H Do, and Zoran Nenadic. 2023. Power Budget of a Skull Unit in a Fully-Implantable Brain-Computer Interface: Bio-Heat Model. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 31 (2023), 4029–4039.

[74] Krishna Shenoy and Jonathan Kao. 2021. Measurement, manipulation and modeling of brain-wide neural population dynamics. *Nature Communications* 12 (12 2021). https://doi.org/10.1038/s41467-020-20371-1

[75] Robert K. Shepherd. 2016. *Neurobionics: The biomedical engineering of neural prostheses.* John Wiley & Sons.

[76] Takanori Shinotsuka, Yasuhiro R. Tanaka, Shin-Ichiro Terada, Natsuki Hatano, and Masanori Matsuzaki. 2023. Layer 5 Intratelencephalic Neurons in the Motor Cortex Stably Encode Skilled Movement. *Journal of Neuroscience* 43, 43 (2023), 7130–7148. https://doi.org/10.1523/JNEUROSCI.0428-23.2023 arXiv:https://www.jneurosci.org/content/43/43/7130.full.pdf

[77] Elliot Smith, Guillermo Horga, Mark Yates, Charles Mikell, Garrett Banks, Yagna Pathak, Catherine Schevon, Guy McKhann, Benjamin Hayden, Matthew Botvinick, and Sameer Sheth. 2019. Widespread temporal coding of cognitive control in the human prefrontal cortex. *Nature Neuroscience* 22 (11 2019), 1–9. https://doi.org/10.1038/s41593-019-0494-0

[78] Nicholas A. Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, Susu Chen, Jennifer Colonell, Richard J. Gardner, Bill Karsh, Dimitar Kostadinov, Carolina Mora-Lopez, Junchol Park, Jan Putzeys, Britton Sauerbrei, Rik J. J. van Daal, Abraham Z. Vollan, Marleen Welkenhuysen, Zhiwen Ye, Joshua Dudman, Barundeb Dutta, Adam W. Hantman, Kenneth D. Harris, Albert K. Lee, Edvard I. Moser, John O'Keefe, Alfonso Renart, Karel Svoboda, Michael Häusser, Sebastian Haesler, Matteo Carandini, and Timothy D. Harris. 2020. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *bioRxiv* (2020). https://doi.org/10.1101/2020.10.27.358291

[79] Ian Stevenson and Konrad Kording. 2011. How advances in neural recording affect data analysis. *Nature neuroscience* 14 (02 2011), 139–42. https://doi.org/10.1038/nn.2731

[80] Aaron Stillmaker and Bevan Baas. 2017. Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm. *Integration* 58 (2017), 74–81. https://doi.org/10.1016/j.vlsi.2017.02.002

[81] Aneesha K. Suresh, James M. Goodman, Elizaveta V. Okorokova, Matthew Kaufman, Nicholas G. Hatsopoulos, and Sliman J. Bensmaia. 2020. Neural population dynamics in motor cortex are different for reach and grasp. *eLife* 9 (11 2020), e58848. https://doi.org/10.7554/eLife.58848

[82] Sonia Todorova, Patrick Sadtler, Aaron Batista, Steven Chase, and Valérie Ventura. 2014. To sort or not to sort: The impact of spike-sorting on neural decoding performance. *Journal of Neural Engineering*

11, 5 (2014). https://doi.org/10.1088/1741-2560/11/5/056005

[83] Eric M. Trautmann, Sergey D. Stavisky, Subhaneil Lahiri, Katherine C. Ames, Matthew T. Kaufman, Daniel J. O'Shea, Saurabh Vyas, Xulu Sun, Stephen I. Ryu, Surya Ganguli, and Krishna V. Shenoy. 2019. Accurate Estimation of Neural Population Dynamics without Spike Sorting. *Neuron* 103, 2 (2019), 292–308.e4. https://doi.org/10.1016/j.neuron.2019.05.003

[84] Shreejoy J. Tripathy, Judith Savitskaya, Shawn D. Burton, Nathaniel N. Urban, and Richard C. Gerkin. 2014. NeuroElectro: a window to the world's neuron electrophysiology data. *Frontiers in Neuroinformatics* 8 (2014). https://doi.org/10.3389/fninf.2014.00040

[85] Wilson Truccolo, Jacob A. Donoghue, Leigh R. Hochberg, Emad N. Eskandar, Joseph R. Madsen, William S. Anderson, Emery N. Brown, Eric Halgren, and Sydney S. Cash. 2011. Single-neuron dynamics in human focal epilepsy. *Nature Neuroscience* 14, 5 (2011), 635–643. https://doi.org/10.1038/nn.2782

[86] Alexandra V. Ulyanova, Carlo Cottone, Christopher D. Adam, Kimberly G. Gagnon, D. Kacy Cullen, Tahl Holtzman, Brian G. Jamieson, Paul F. Koch, H. Isaac Chen, Victoria E. Johnson, and John A. Wolf. 2019. Multichannel silicon probes for awake hippocampal recordings in large animals. *Frontiers in Neuroscience* 13, APR (2019), 1–16. https://doi.org/10.3389/fnins.2019.00397

[87] Daniel Valencia and Amirhossein Alimohammad. 2019. An Efficient Hardware Architecture for Template Matching-Based Spike Sorting. *IEEE Transactions on Biomedical Circuits and Systems* 13, 3 (2019), 481–492. https://doi.org/10.1109/TBCAS.2019.2907882

[88] Rakesh Veerabhadrappa, Masood Ul Hassan, James Zhang, and Asim Bhatti. 2020. Compatibility Evaluation of Clustering Algorithms for Contemporary Extracellular Neural Spike Sorting. *Frontiers in Systems Neuroscience* 14 (2020), 1–17. Issue June. https://doi.org/10.3389/fnsys.2020.00034

[89] Valérie Ventura. 2008. Spike train decoding without spike sorting. *Neural Computation* 20, 4 (2008), 923–963. https://doi.org/10.1162/neco.2008.02-07-478

[90] Rio J. Vetter, Justin C. Williams, Jamille F. Hetke, Elizabeth A. Nunamaker, and Daryl R. Kipke. 2004. Chronic neural recording using silicon-substrate microelectrode arrays implanted in cerebral cortex. *IEEE Transactions on Biomedical Engineering* 51, 6 (2004), 896–904. https://doi.org/10.1109/TBME.2004.826680

[91] Ziqiang Wei, Bei-Jung Lin, Tsai-Wen Chen, Kayvon Daie, Karel Svoboda, and Shaul Druckmann. 2019. A comparison of neuronal population dynamics measured with calcium imaging and electrophysiology. *bioRxiv* (2019). https://doi.org/10.1101/840686

[92] Di Wu, Jingjie Li, Zhewen Pan, Younghyun Kim, and Joshua San Miguel. 2022. UBrain: A Unary Brain Computer Interface. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 468–481. https://doi.org/10.1145/3470496.3527401

[93] Yuning Yang and Andrew J. Mason. 2017. Hardware Efficient Automatic Thresholding for NEO-Based Neural Spike Detection. *IEEE Transactions on Biomedical Engineering* 64, 4 (2017), 826–833. https://doi.org/10.1109/TBME.2016.2580319

[94] Pierre Yger, Giulia L.B. Spampinato, Elric Esposito, Baptiste Lefebvre, Stéphane Deny, Christophe Gardella, Marcel Stimberg, Florian Jetter, Guenther Zeck, Serge Picaud, Jens Duebel, and Olivier Marre. 2018. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *eLife* 7 (2018), 1–23. https://doi.org/10.7554/eLife.34518

[95] Xinyue Yuan, Manuel Schröter, Marie Engelene J. Obien, Michele Fiscella, Wei Gong, Tetsuhiro Kikuchi, Aoi Odawara, Shuhei Noji, Ikuro Suzuki, Jun Takahashi, Andreas Hierlemann, and Urs Frey. 2020. Versatile live-cell activity analysis platform for characterization of neuronal dynamics at single-cell and network level. *Nature Communications* 11 (12 2020). Issue 1. https://doi.org/10.1038/s41467-020-18620-4

[96] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. 2020. GOBO: Quantizing Attention-Based NLP Models for Low Latency and Energy Efficient Inference. *Proceedings of the Annual International Symposium on Microarchitecture, MICRO* 2020-October (2020), 811–824. https://doi.org/10.1109/MICRO50266.2020.00071 arXiv:2005.03842

[97] Christopher A. Zimmerman, Alejandro Pan-Vasquez, Bichan Wu, Emma F. Keppler, Eartha Mae Guthman, Robert N. Fetcho, Scott S. Bolkan, Brenna McMannon, Junuk Lee, Austin T. Hoag, Laura A. Lynch, Sanjeev N. Janarthana, Juan F. López Luna, Adrian G. Bondy, Annegret L. Falkner, Samuel S-H Wang, and Ilana B. Witten. 2023. A neural mechanism for learning from delayed postingestive feedback. (10 2023). https://doi.org/doi:10.1101/2023.10.06.561214 Preprint at webpage https://pubmed.ncbi.nlm.nih.gov/37873112/.