Ultra Low Power, Event-Driven Data Compression of Multi-Unit Activity

Oscar W. Savolainen, Zheng Zhang, Timothy G. Constandinou

November 24, 2022

Abstract

Recent years have demonstrated the feasibility of using intracortical Brain-Machine Interfaces (iBMIs), by decoding thoughts, for communication and cursor control tasks. iBMIs are increasingly becoming wireless due to the risk of infection and mechanical failure, typically associated with percutaneous connections. The wireless communication itself, however, increases the power consumption further; with the total dissipation being strictly limited due to safety heating limits of cortical tissue. Since wireless power is typically proportional to the communication bandwidth, the output Bit Rate (BR) must be minimised. Whilst most iBMIs utilise Multi-Unit activity (MUA), i.e. spike events, and this in itself significantly reduces the output BR (compared to raw data), it still limits the scalability (number of channels) that can be achieved. As such, additional compression for MUA signals are essential for fully-implantable, high-information-bandwidth systems. To meet this need, this work proposes various hardware-efficient, ultra-low power MUA compression schemes. We investigate them in terms of their BRs and hardware requirements as a function of various on-implant conditions such as MUA Binning Period (BP) and number of channels. It was found that for BPs ≤ 10 ms, the delta-asynchronous method had the lowest total power and reduced the BR by almost an order of magnitude relative to classical methods (e.g. to approx. 151 bps/channel for a BP of 1 ms and 1000 channels on-implant.). However, at larger BPs the synchronous method performed best (e.g. approx. 29 bps/channel for a BP of 50 ms, independent of channel count). As such, this work can guide the choice of MUA data compression scheme for BMI applications, where the BR can be significantly reduced in hardware efficient ways. This enables the next generation of wireless iBMIs, with small implant sizes, high channel counts, low-power, and small hardware footprint. All code and results have been made publicly available.

1 Introduction

1.1 Wireless Intracortical Brain-Machine Interfaces

Brain machine interfaces (BMIs) are electronic devices that measure neural activity, extract informative features-features from that activity, and convert those features into outputs that replace, restore, enhance, supplement, or improve human functions. They can, for example, be used to treat paraplegia, quadriplegia, movement disorders, Locked-in syndrome and more [**?**, 1, 2]. Intracortical BMIs (iBMIs) are the most invasive form of BMI, where electrodes are placed into brain tissue [3]. They also provide the highest resolution of BMI data, capable of measuring the firing times of individual neurons in the electrodes' vicinity.

For iBMIs, wireless communication and powering is becoming increasingly essential [3–5], particularly for translational efforts. This eliminates physical percutaneous connections, e.g. wires breaching the skin, and associated risks (infection, mechanical damage, etc), while significantly improving the quality of life of the user. They also enable chronic powering without having to replace the implant as battery life is limited. The move to wireless iBMIs (WI-BMI) is required for the effective translation of BMIs into clinical applications and is one of the remaining bottlenecks. WI-BMIs consist of a small implant in the brain that communicates wirelessly with an external decoder, where more computationally demanding processing and/or decoding is performed. There is an additional desire to minimize the invasiveness of WI-BMIs through reducing minimize the size of the implant to the mm or sub-mm scale, so as to minimize damage and foreign body response in the brain [3].

1.2 Multi-Unit Activity

The most common signal in modern iBMIs is the Multi-Unit Activity (MUA) signal. When a neuron fires in the vicinity of an intracortical electrode, the electrode measures a short spike in voltage. In the MUA signal, the timing of this spike is noted, and the firing times binned at some resolution, i.e. the Binning period (BP). Therefore, at each

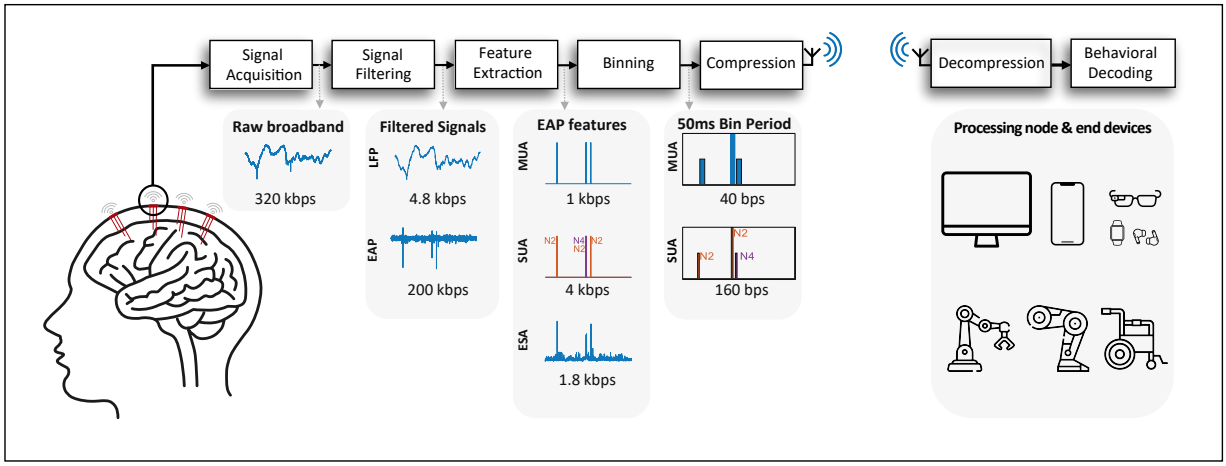

Figure 1: Reprinted from [13] with permission. Typical BMI data processing and compression flow, with common extracted features / lossy compressions of intracortical broadband data. The numerical values beneath the signals give approximate BRs per channel for that signal. LFP: Local Field Potential, equal to the lowpassed broadband at approx. 300 Hz. EAP: Extracellular Action Potential, equal to the highpassed broadband at approx. 300 Hz. MUA: Multi-Unit Activity, equal to the unsorted thresholded spike activity from the EAP. SUA: Single-Unit Activity, equal to the sorted thresholded spike activity from the EAP. ESA: Entire Spiking Activity, equal to the enveloped rectified EAP signal, giving an envelope of unsorted spiking activity.

BP, each electrode channel outputs an integer value representing the MUA Firing Rate (FR), equal to how many spikes were detected by the electrode during the BP. The MUA signal has been found to have high Behavioral Decoding Performance (BDP) [6–9] while being simple to extract on-implant [10–12], adding to its popularity.

The MUA BP is typically 1 ms [5], since spikes last approximately 2 ms. The dynamic range (S) is also typically set to a binary output ($S = 2$), giving $ceil(log_2(2)) = 1$ bit/sample, which all together corresponds to a bit rate (BR) of 1 kbps/channel. This is a significant reduction over the raw broadband signal bandwidth, which is typically sampled at some 16 bits/sample at 20 kHz, giving a broadband BR of approximately 320 kbps/channel. However, a 1kbs/channel is still too high for many WI-BMI applications, discussed in the next section.

For the WI-BMI environment, the processing flow for the MUA signal, along with that of the other most common intracortical neural features and their typical BRs, is given in Fig. 1.

1.3 Heating in WI-BMIs

Heating of cortical tissue is a major issue with WI-BMIs. This is because heating tissue can cause irreparable damage [14]. However, the acceptable heat limit in cortical tissue is not robustly understood [13,14]. Additionally, any power that is used on-implant needs to be transferred wirelessly to the implant, some large portion of which is absorbed by the neural medium, heating it [14]. Therefore, on-implant power should be strictly limited. A 10 mW/cm^2 heat flux is perhaps a conservative upper safety limit for cortical tissue [13], when accounting for wireless power transfer effects.

For an example 2.5 $\times$ 2.5 mm scale implant, this gives an upper power budget of 625 μW. If the entirety of the available implant power budget goes into communication and one has a state-of-the-art FPGA-based RF-backscatter WI-BMI communication energy of 20 nJ/bit [13, 15], this produces a maximum BR of 31.3 kbits/s. Therefore, at 1 kbits/s, this implant could only transmit up to 31 channels if all of the power was used for communication and we had an exceptionally low communication energy per bit. In practice, reported communication energies are typically not measured *in* or *ex vivo*, and the communication energy of the same system in the neural medium would be significantly higher. For example, the lowest FPGA communication energy that had been shown was 20 nJ/bit [13, 15], but this involved a head-mounted wireless communication board external to the cranium of the free-moving rat subject. As such, with practically higher communication energies, the number of channels that can be fit on-implant is fewer.

Additionally, one has to account for the hardware static power, where even ultra-low power FPGAs like the Lattice ice40LP1K require over 100μW for the static power consumption. Furthermore, we need to account for the MUA extraction power, the front-end pre-amplifier and ADC power costs [13]. As such, it seems unlikely that a 2.5 $\times$ 2.5 mm scale FPGA implant could host more than 10 MUA channels on-implant at 1 kbps/channel BRs while staying

Table 1: Huffman encoder example [13]. For the given frequencies and codeword lengths, the compressed data has an average length of 0.8 × 1 + 0.1 × 2 + 0.07 × 3 + 0.03 × 3 = 1.4 bits, versus the 2 bits that are normally required to represent 4 symbols with a fixed-length encoding.

Huffman Encoder		
Symbol / Key	**Frequency**	**Codeword**
0	0.8	0
1	0.1	10
2	0.07	110
3	0.03	111

within heating limits. An ASIC implant might be able to, however, some form of data compression is required to enable greater channel counts in the next generation of mm and sub-mm scale WI-BMIs, even for low-bandwidth signals like MUA.

1.4 Data Compression in WI-BMIs

Data compression comes in two forms: lossless and lossy. Lossless compression consists of giving shorter codewords to more common signals, which is the basis of Information Theory [16]. There are many lossless compression algorithms, however most of them require too many resources to be practical for extremely hardware-constrained devices like WI-BMIs, e.g. Lempel-Ziv, Arithmetic, or Adaptive Huffman encoding. A lossless compression algorithm that works well for WI-BMIs is Static Huffman (SH) encoding, since it can be implemented using only a small number of Look-Up Tables (LUT). It works by one pre-training a Huffman encoder [17] on some representative data, and then implementing the encoder on-implant with LUTs. If the data that is measured on-implant is well-represented by the training data, the encoder will efficiently compress the data. An example of a Huffman encoder is shown in Table 1. SH encoders were used to great effect in [13] to compress MUA signals. They were also used in [18] to compress other intracortical signals, e.g. Entire Spiking Activity, Local Field Potential and Extracellular Action Potentials. Huffman encoders are optimal among methods that encode symbols separately. Given their hardware efficiency, in their static form, they are attractive for WI-BMIs.

The second form of data compression is lossy compression. One eliminates information that is assumed to not be of interest to the final application, reducing the bandwidth. This is also called feature extraction, and an example is to go from the raw broadband signal to the MUA signal. However, the MUA signal can also be further lossily compressed from its standard binary dynamic range and 1 ms BP. In [5, 19], it was proposed that increasing the BP would be an effective form of MUA data compression. This was shown empirically in [13]. They found that increasing MUA BP up to 100 ms had no negative impact on BDP, which is the final output of interest for motory BMIs, while reducing the communication power significantly. Another aspect is that when BP is increased, the resulting dynamic range of the signal can grow. For example, a 100 ms BP is likely to have far more spike events detected in a bin than a 1 ms BP. However, [13] found that one could significantly limit the dynamic range S, lossily compressing the data, without reducing the BDP. This is done by taking all MUA FR values above $S - 1$, and setting them to $S - 1$. As such, there are two simple ways to lossily compress MUA data: increasing BP and reducing S.

1.5 This Work

A hardware-efficient MUA compression system was introduced in [13]. At its base, it used binning, limiting the dynamic range and SH encoding. Additional techniques were also considered, such as the use of sample histograms for SH mapping and the use of multiple SH encoders for channel assignment. It achieved state-of-the-art compression performance, especially when the BP was 50 ms or above. However, when the BP was smaller, even though ×2-3 bandwidth reduction was achieved relative to standard MUA representation, the absolute bandwidth was still too much for wireless transmission. Additionally, there is a desire for finer temporal resolution in BMIs, to enable a faster control loop and improve user experience. As such, some form of MUA compression, tailored to smaller BPs, is desired.

Shorter BPs make the firing rates sparse. In other words, in a smaller time window it is more likely that a FR of 0 will occur. Using this known property of MUA signals can help us improve the compression performance for BPs below 50ms.

In this work, we reproduced the work in [13] as a baseline method. This method is referred to as the 'synchronous' architecture, where each channel outputs its data for each BP. We also propose several 'asynchronous'

architectures, where only data from active channels is transmitted for the given BP, taking advantage of sparse MUA FRs at lower BPs. Overall, four compression schemes will be examined: the synchronous scheme from [13], an explicit asynchronous method, which transmits the active channel IDs with corresponding spike counts, a delta-asynchronous method, which transmits the delta-sampled active channel IDs with corresponding spike counts, and a group asynchronous method which uses a form of run-length encoding.

1.6 Original Contributions of This Work

This paper contains multiple original contributions to MUA compression. These are as follows:

- Design and analysis of novel MUA data compression architectures, such as the explicit asynchronous, delta-asynchronous, and group asynchronous encodings.
- The hardware designs, made publicly available, of these encodings.
- Comparison of these encodings to the state of the art in MUA compression. Very large improvements were made in terms of communication bandwidth for BPs below 20 ms, appropriate for higher temporal resolution MUA-based BMIs. Compression performance was improved by up to over an order of magnitude for smaller BPs relative to the state of the art. This allows for significantly more MUA channels to be fit on-implant while staying within heating limits. This was found to be true using data from 3 different publicly available datasets, including 5 non-human primate subjects and 23 hours of Utah array multi-channel recordings.
- We showed that Static Huffman encoders, trained on distributions that make simple assumptions about MUA data, perform just as well as non-causal Adaptive Huffman encoders while being far more hardware-efficient and causal. As such, SH encoders seem to be a very attractive means of data compression for MUA-based WI-BMIs.
- We investigated the use of sample histograms for mapping, as in [13], for the asynchronous schemes proposed in this work. We determined their effect on BR and processing power, as well as analyzing trade-offs in their use.

The rest of this paper is organised as follows: Section 2 introduces three datasets used in this work and details the compression methods. Section 3 shows the compression performance and hardware cost of each method. Based on these results, we give the recommend compression settings at different BP and channel counts, trading off between the BR, processing power and hardware cost. Section 4 discusses some algorithm and hardware design considerations in MUA compression and Section 5 concludes this work.

2 Materials and Methods

All compression algorithm work was done in MATLAB R2021A. All hardware design and optimisation work was done using ModelSim Lattic Edition and IceCube2020.12. All code and results are publicly available at [20].

2.1 Dataset and Data Formatting

Brain signal conditions can vary across subjects and tasks. In order to reduced the bias, we have used three different publicly available datasets [21–23], summarised in Table. 2. Further details are given in Supplemental Material, **Section XX**. For each dataset, the SUA data was intra-channel collated to MUA, then binned to the desired BP, where BP $\in [1, 5, 10, 20, 50, 100]$ ms. Based on the results in [13], we limited S to [2, 2, 2, 2, 3, 5] for BPs of [1, 5, 10, 20, 50, 100] ms respectively. The Supplemental Material includes results from S values $\in [\mathbb{Z}^+, 2, 3, ..., 14, 15]$.

The length of the recordings was largely irrelevant for the sake of this work, as all encoding was done without using a time derivative and the bandwidth was measured in [bps/channel], normalising for time. As such, a standard length of 100 s was set for each recording, long enough to gather a stable distribution for each channel for any of the tested BPs. To maximize the use of the data in observing the effect of channel count, recordings were split into consecutive 100 s segments and collated together. For example, a 400 s recording was represented as 4 parallel channels of 100 s long recordings, where x_o,...,$x_{N/4-1}$ became one channel, $x_{N/4}$,...,$x_{N/2-1}$ became the next, etc., where x_k is a single-channel MUA recording and $N = 400\,s/\text{BP}$ is the length of the recording in samples. A total of 79200 channels were available after splitting and collation.

The data was then split into training and testing sets. This is because the encodings have parameters that need to be optimised, which was done on the training set. The testing of parameter-optimised encoding on the testing

Table 2: Dataset summaries.

Dataset	Neural data type	Species, electrode type and brain region	Details	Behaviour
Flint [21]	SUA	Rhesus macaque monkey Utah array M1	One subject 12 recordings across 5 days 96 channels Recording lengths (quartiles, s): 597, 604, 630	Free-reaching hand task Continuous data stored
Sabes [22]	SUA	Rhesus macaque monkeys Utah array M1 and S1	Subjects Indy and Loco 37 recordings for Indy across 10 months 10 recordings for Loco across a month 96-192 channels Recording lengths (quartiles, s): Indy: 472, 524, 816 Loco: 1771, 1928, 2384	Free-reaching hand task Continuous data stored
Brochier [23]	SUA	Rhesus macaque monkeys Utah array M1 and PMv/PMd	Subjects N and L Single session recordings 96 channels Recording lengths (s): N: 1003 L: 709	Hand reaching task Target stored

set can then provide an objective estimation of the encoding performance on the unseen data. The training-testing split was done by randomly selecting 30000 channels and placing them into the training set. The remaining 49200 were put into the testing set. For each encoding, BP, S and n combination, the training and testing results were each averaged across 5 training and testing runs. For each run, $n \in [10, 100, 1000, 10000, 30000]$ channels were selected randomly without replacement from amongst all training or testing channels.

2.2 Synchronous encoding

The synchronous encoding [13] was the first scheme to be implemented. It is called "synchronous" because it always sends out a symbol for each channel, regardless of whether an event occurs on a channel or not. It consists of representing the multi-channel MUA data as multiplexed data, and is a form of time-division multiple access (TDMA) encoding, where only 1 symbol is communicated per channel per time-division i.e. BP. Without lossless compression, the length of the data block is $n \times m$, where n is the number of channels and m is the number of bits used to represent the number of MUA events per BP on each channel. An example using a standard binary representation with m and $n = 3$:

$$001\ 111\ 000$$

would indicate that 1 neuron fired (001) on channel 1, 7 neuron firings occurred (111) on channel 2, and 0 (000) on channel 3. An advantage is that the channel ID is implicitly encoded in bit position, and so does not need to be explicitly encoded. E.g.,

$$c = ceil(t/m) \tag{1}$$

where $c \in [\mathbb{Z},\ 1 \leq c \leq n]$ is the channel ID, $t \in [\mathbb{Z},\ 1 \leq t \leq n \times m]$ is the bit position and $ceil$ is the ceiling function. m is the number of bits required to represent all possible MUA FRs losslessly, and is generally set as $ceil(log_2(\texttt{max}(X) + 1))$, where X is the multi-channel MUA data. However, as discussed in Section 1.4, one can lossily compress the data by limiting the dynamic range at an S value, setting $X[X > (S-1)] \leftarrow (S-1)$. It requires

$$m' = ceil(log_2(S)) \tag{2}$$

bits to represent a range of 0 to $S-1$ MUA FRs. If $m' < m = ceil(log_2(\texttt{max}(X)+1))$, this lossily reduces the required bandwidth. As such, the synchronous encoding typically has a BR of:

$$\mathrm{BR} = \frac{m'}{\mathrm{BP}} \qquad \text{[bps/channel]} \quad (3)$$

However, as in [13] we integrated the synchronous encoding with a SH encoder that compressed the FRs. The SH encoder was pre-trained on a decaying exponential which mimics the distribution of MUA FRs at BPs ≤ 100 ms [13], where smaller codewords are given to smaller FRs.

2.3 Explicit Asynchronous encoding

In the synchronous architecture, the FR of each channel is encoded. The channel ID is implicitly encoded in bit position. To the best of the authors' knowledge, the following asynchronous architectures are proposed for the first time in MUA compression. In these asynchronous architectures, the channel ID is explicitly encoded and only the activate channel will be transmitted. i.e., the channel ID is only sent out if a non-zero FR occurs on that channel, followed by a binary codeword representing the rate. If a channel has a FR of 0, nothing gets communicated for that channel, and the offline decoder assumes the missing channels had FRs of 0. For sparse signals, i.e. where FRs not equal to 0 are rare, this can offer reduced bandwidth over the synchronous paradigm, where even FRs of 0 need to be communicated for each channel.

As the channel IDs need to be explicitly encoded, the simplest method is to give the channels a standard binary codeword of length k_1, where:

$$k_1 = ceil(log_2(n)) \qquad \text{[bits]} \quad (4)$$

Similarly, the MUA FR for each channel is encoded as a binary codeword of length m_2:

$$m_2 = ceil(log_2(S-1)) \qquad \text{[bits]} \quad (5)$$

An exception occurs if $S = 2$, i.e. $m_2 = 1$ and i is limited to 0 and 1. In that case the firing rate codeword is unnecessary as the decoder can assume all received channel has the firing rate of 1. An example of the encoding without lossless compression, with $n = 4$, $k_1 = 2$ and $m_2 = 2$, is given by:

0001 0100 1011

This shows that channel 1 (00) had a FR of 2 events (01) in the bin, channel 2 (01) had FR = 1 (00), channel 3 (10) had FR = 4 (11), and channel 4 had a FR = 0 (absent). In this work, this encoding was named the explicit asynchronous encoding because the FR per channel is explicitly encoded in the m_2-length codeword that follows the channel ID.

To include the SH encoder, there was no good way to compress the channel IDs, since they are *a priori* equally likely to be active. Therefore, the explicit asynchronous encoding uses the same k_1 length codeword for the channel IDs, but uses varying length SH codewords for the FRs. As in the SH synchronous implementation, the FR encoder was trained on a decaying exponential.

2.4 Delta-Asynchronous encoding

The previous encoding suffers since the probability of each channel being active is *a priori* equal. As such, the channel IDs cannot be compressed as they are with SH encoders. However, the channel difference between two successive active channels has varying probability, and therefore Huffman encoding can be used. In other words, we can encode the delta-sampled channel IDs of active channels. Therefore, in this 'delta-asynchronous' encoding, if a channel has a FR above 0, it is given a Δ value by subtracting its ID, $j_{current}$, from the ID of the previous channel to have a FR above 0, $j_{previous}$. I.e., $\Delta = j_{current} - j_{previous}$.

Therefore, the FRs were compressed with the same SH encoder as in the explicit asynchronous encoding. However, the Δ-sampled channel IDs were compressed using a SH encoder trained on a decaying exponential. Significant memory optimisation was also done by setting a maximum Δ-sampled SH encoder size, using a form of run-length encoding. This reduced BR slightly but significantly reduced memory requirements. The details of this optimisation are extensively detailed in Sections 3 and 4 in the Supplemental Material.

2.5 Group Asynchronous encoding

In the group asynchronous encoding, one uses position and stop symbols to encode the FRs. As in the explicit asynchronous encoding, one explicitly encodes the channel ID, but here one encodes the FR per channel implicitly in channel ID position. For example, in decimal,

$$2\,4\,stop_2\,1\,6\,stop_3\,stop_4\,3$$

signifies that channels 2 and 4 had a FR of $i = 1$ in the given bin, channels 1 and 6 had a FR of $i = 2$, channel 3 had FR $i = 4$, and the rest of the channels had FR $i = 0$. For the codeword lengths, the simplest implementation is to give the stop symbols and the channel IDs a length of k_2 bits, where:

$$k_2 = ceil(log_2(n + S - 2)) \qquad \text{[bits]} \quad (6)$$

E.g. $n = 2$, $S = 4$, means that there are 2 channels, FRs between 0 and 3 inclusive can be encoded, and $k_2 = 2$. Channel 1 gets a codeword of 00, channel 2 gets a codeword 01, and the stop symbols for $i = 2$ and $i = 3$ get codewords of 10 and 11 respectively. It essentially involves sorting the channels by FR, and using a form of run-length encoding.

There was no clear role for SH encoding, since the stop symbols and FR codewords need to come from the same encoder. Estimating these probabilities *a priori* is difficult. As such, no SH encoding for the group encoding was used.

2.6 Hardware Implementation

The above encoding algorithms were implemented on an ultra-low power, hardware-efficient FPGA target, Lattice ice40LP1K, in order to assess their hardware complexity (power consumption and resources usage). The results are a good approximation of the ultimate ASIC design, but enables rapid testing of algorithm parameters. The implementations are given in the Supplemental Material, Section 3.

3 Results

In Fig. 2, we plotted the BRs of each encoding at different BPs and n. We can observe that the synchronous and delta-asynchronous encodings perform best, with the best one depending on BP and n. From the hardware perspective, the synchronous encoding is far more hardware efficient than the delta-asynchronous encoding. Whilst the amount of logic cells is similar for the two encodings, the delta-asynchronous encoding uses significantly more memory, and therefore the processing power is higher. Additionally, the synchronous scheme performs the same independent of channel count. However, the delta-asynchronous scheme is affected by channel count. When n increases, the delta-sampled channel IDs can be larger, meaning longer codewords and higher BRs. As such, increasing n slightly increase the delta-asynchronous BR, but significantly less than for the explicit and group asynchronous encodings.

As such, the optimal selection generally varies as a function of channel count and BP. Ultimately, we want the total power on-implant to be reduced. As such, Fig. 3 shows the total dynamic power for the FPGA implementation, made up of the processing power for each encoding and the communication power, estimated as the BR multiplied by an estimated 20 nJ/bit communication energy.

3.1 Optimal Encoding Selection

From Fig. 3 and our analysis of the hardware costs in the Supplemental Material, for each BP and n we selected the optimal compression system. These are given in Table 3 (a). We then determined each selected system's performance on the test data, i.e. data that had hereto been untouched, and the BRs are given in (b). The associated required number of FPGA logic cells and memory are given respectively in Table 3 (c) and (d). The total dynamic power on the Lattice ice40LP FPGA target, determined on the testing data, is given in (e). In summary, the delta-asynchronous encoding is suitable for short BPs (less than approx. 20 ms) while the synchronous encoding is preferred when the channel count or BP is increased.

3.2 Testing the Encoding/Decoding

While having the BRs for each encoding and implementation is important, it is necessary to verify that each encoding functions as desired and is feasible. As such, encoding functions that encoded the binned and S-saturated MUA

Table 3: Suggested static encodings and corresponding BRs and hardware resources for each BP and channel count. BR results from test data. 'S' represents the synchronous encoding. 'DA' represents the delta-asynchronous encoding. 'x' entries are where the memory requirements exceeded those of the FPGA platform we selected. The FPGA Logic Cells and Memory are shared across all channels.

(a) Recommended Encoding

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	DA	DA	DA	DA	DA
5	**2**	DA	DA	DA	DA	DA
10	**2**	DA	DA	DA	DA	DA
20	**2**	DA	S	S	S	S
50	**3**	DA	S	S	S	S
100	**5**	S	S	S	S	S

(b) Bit Rates (bps / channel)

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	50	99	151	177	183
5	**2**	45	75	78	79	79
10	**2**	44	62	62	62	62
20	**2**	31	50	50	50	50
50	**3**	25	29	29	29	29
100	**5**	20	21	22	22	22

(c) FPGA Logic Cells

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	114	144	164	207	242
5	**2**	124	153	174	217	252
10	**2**	130	159	180	223	258
20	**2**	135	128	134	162	192
50	**3**	183	171	176	207	x
100	**5**	232	246	251	286	x

(d) FPGA Memory (bits)

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	60	648	2448	20448	60448
5	**2**	60	968	2768	20768	60768
10	**2**	60	968	2768	20768	60768
20	**2**	60	200	2000	20000	60000
50	**3**	140	400	4000	40000	x
100	**5**	60	600	6000	60000	x

(e) FPGA Dynamic Power (μW / channel)

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	2.03	3.05	4.1	4.74	5.39
5	**2**	1.93	2.58	2.66	2.76	3.34
10	**2**	1.95	2.32	2.35	2.42	2.97
20	**2**	1.7	1.96	1.98	2.03	2.55
50	**3**	1.59	1.54	1.58	2.07	x
100	**5**	1.4	1.43	1.46	2	x

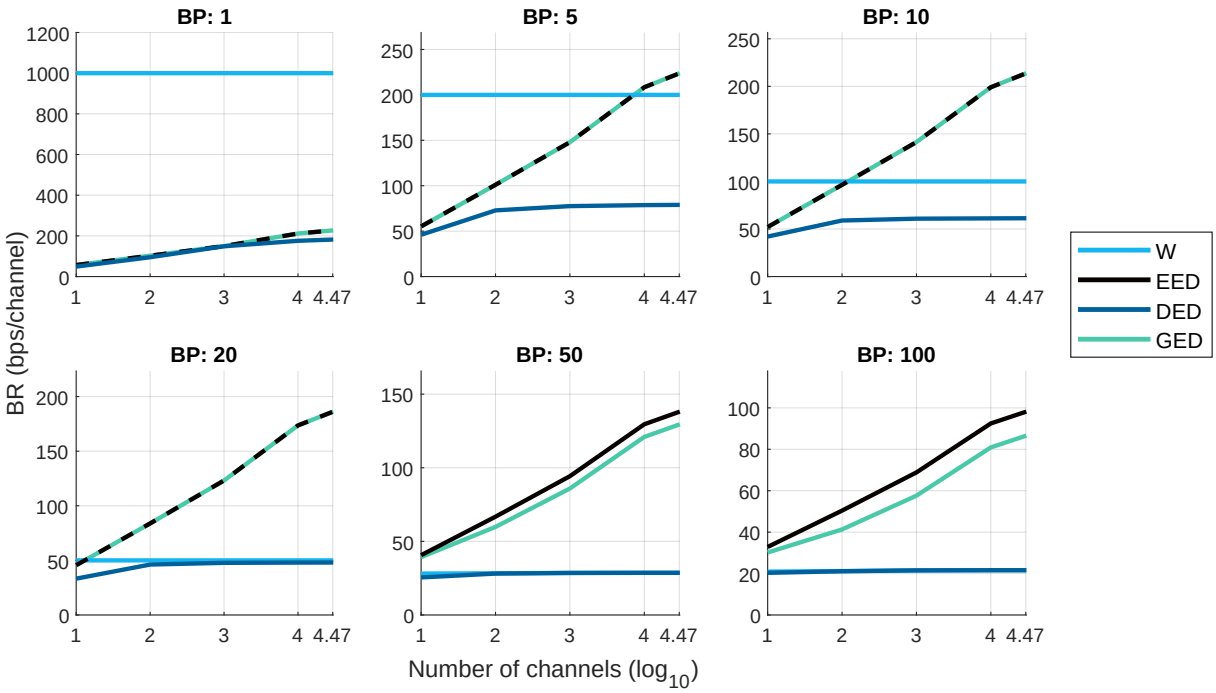

Figure 2: Bit Rates for communication schemes at a BP $\in$ [1, 5, 10, 20, 50, 100] ms. For each BP, S was respectively fixed as [2, 2, 2, 2, 3, 5]. At $S = 2$, the explicit and group asynchronous encodings are mathematically identical, and so their BRs overlap for BPs $\leq$ 20 ms.

data according to each encoding and implementation, as well as with mapped versions, were implemented. The BRs were determined from the encoded data directly and were verified to match the analytically derived BRs, with some very small differences due to rounding errors. The full decodability of the data was then verified by decoding the original binned and S-saturated MUA data from the encodings and verifying that the original and decoded versions were a perfect match. This was done successfully for each method. The MATLAB code for each method and implementation's encoding and decoding is given at [20].

4 Discussion

4.1 Impact of Compression on Channel Counts in FPGA Target

We can take the low-power Lattice ice40LP FPGA board as an example target for our WI-BMI, as in [13]. From this, knowing our BRs per BP and channel count, we can derive how many channels can be hosted on-implant, for different FPGA board dimensions and BPs. We assume a 10 mW/cm^2 heat flux limit and an FPGA static power is 162 μW. We also assume a separate power and hardware budget for the front-end amplifiers, filters and and ADCs, a state-of-the-art communication energy for WI-BMIs on FPGA targets of 20 nJ/bit [13, 15] and a binner processing power of 0.96 μW, independent of BP [13]. Using the information from Table 3 (e), we can derive how many channels can be hosted on-implant while staying within implant power limits. This is compared to the number of channels that can be hosted given the standard uncompressed MUA representation at 1 ms BP, $S = 2$, that has a 1000 kbps/channel BR.

The results are shown graphically in Fig. 4. It can be observed that between 4.7 and 14 times more MUA channels can be fit on-implant with data compression, depending on BP $\in \{1, 5, 10, 20, 50, 100\}$ ms and FPGA size $\in \{2.5, 5, 7.5, 10\}$ mm. As such, one can observe that data compression allows one to fit many more MUA channels onto the same implant. Therefore, it is a useful addition to MUA-based WI-BMIs.

4.2 Effects of Different Firing Rates and Communication Energy

It is of course worth noting that the BRs are taken from 3 datasets that all target the motor cortex during hand kinematic tasks, using Utah arrays. It may be that different tasks, cortical regions and microelectrode arrays will

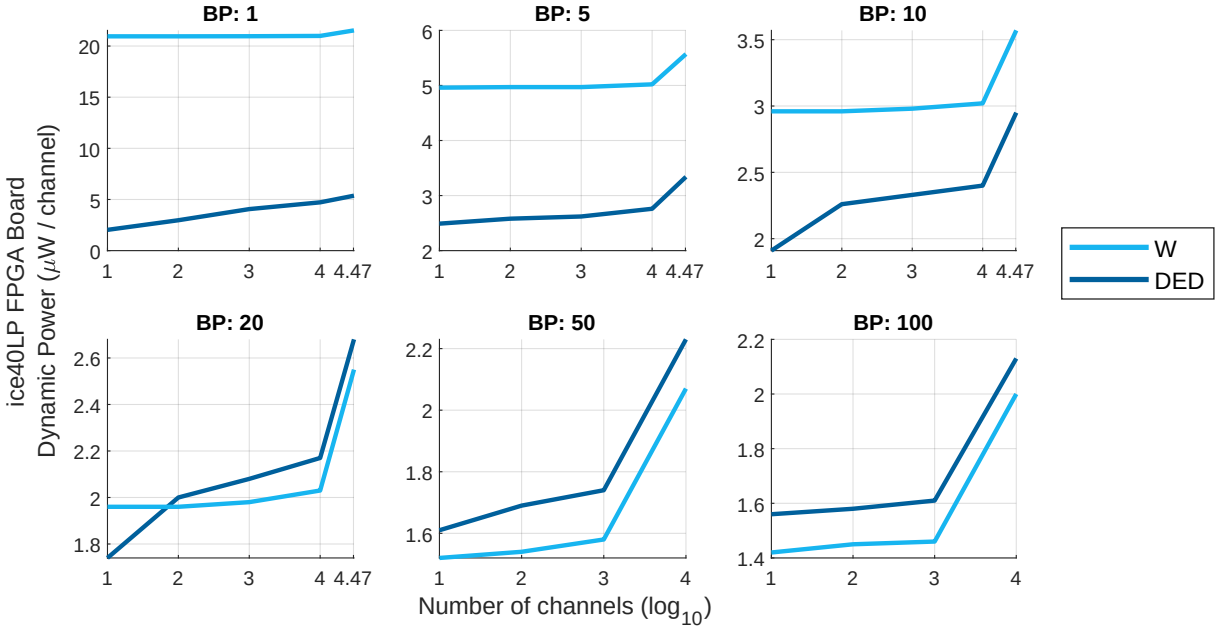

Figure 3: Total communication + processing power, i.e. dynamic power, for the synchronous and delta-asynchronous compression schemes at a BP $\in$ [1, 5, 10, 20, 50, 100] ms. For each BP, S was respectively fixed as [2, 2, 2, 2, 3, 5]. hs was fixed at 0 for each case. For BPs of 50 and 100 ms, the power is not given for 10000 and 30000 channels since the memory requirements exceeded the FPGA target memory budget.

have varying FRs, which will affect the measured BRs. Similarly, a 20 nJ/bit communication energy estimate is used, but this could be overly optimistic. Such values have been observed outside the neural medium, and communication energies within the neural medium are likely to be much higher. As such, while the processing power and resource estimates in this work are robust, the communication energy, made up of the BR and comm. energy per bit, may vary.

Cortical areas, microelectrode arrays and tasks that have higher MUA FRs will be better served, all else held equal, by the synchronous scheme. This is because the synchronous scheme works better if the FRs are non-sparse. However, higher communication energies will make the communication power dominate, and so the importance of the processing power, which is higher in the delta-asynchronous encoding, will be less important when choosing a system. As such, it may be that, even with higher FRs than shown in this work, the delta-asynchronous scheme will be the best option given that it generally has lower BRs than the synchronous scheme. This is especially true at lower BPs, i.e. higher temporal resolutions. Additionally, at higher communication energies, the role of data compression as a whole becomes more important. As such, the ratio of compressed to uncompressed channels that can be fit on-implant within heat limits, shown in Fig. 4, may be a conservative estimate. With significantly higher communication energies, a much larger ratio of compressed channels to uncompressed channels could be fit on-implant.

4.3 Adaptive Huffman vs. Static Huffman encoding

While it was not shown in the main manuscript to simplify it, this work did not just look at SH encoding. For each of the four architecture, we also investigated modes with no lossless compression, with Adaptive Huffman (AH) encoding, which trains the Huffman encoders using the data collected on implant in real time, and the entropic bandwidth, which gives the minimum BRs that could be achieved for each architecture given a perfect compression scheme. These modes are described in detail in the Supplemental Material, and the BR results given. An interesting observation was that the SH encodings performed virtually identically to the AH encodings, which were fully adaptive and assumed perfect knowledge of the to-be-compressed data. As such, we can conclude that SH encoders are a remarkably hardware-efficient and well-performing compression scheme for MUA data. This is because the shape of the to-be-compressed data is more or less known *a priori*, and so on-implant training is unnecessary.

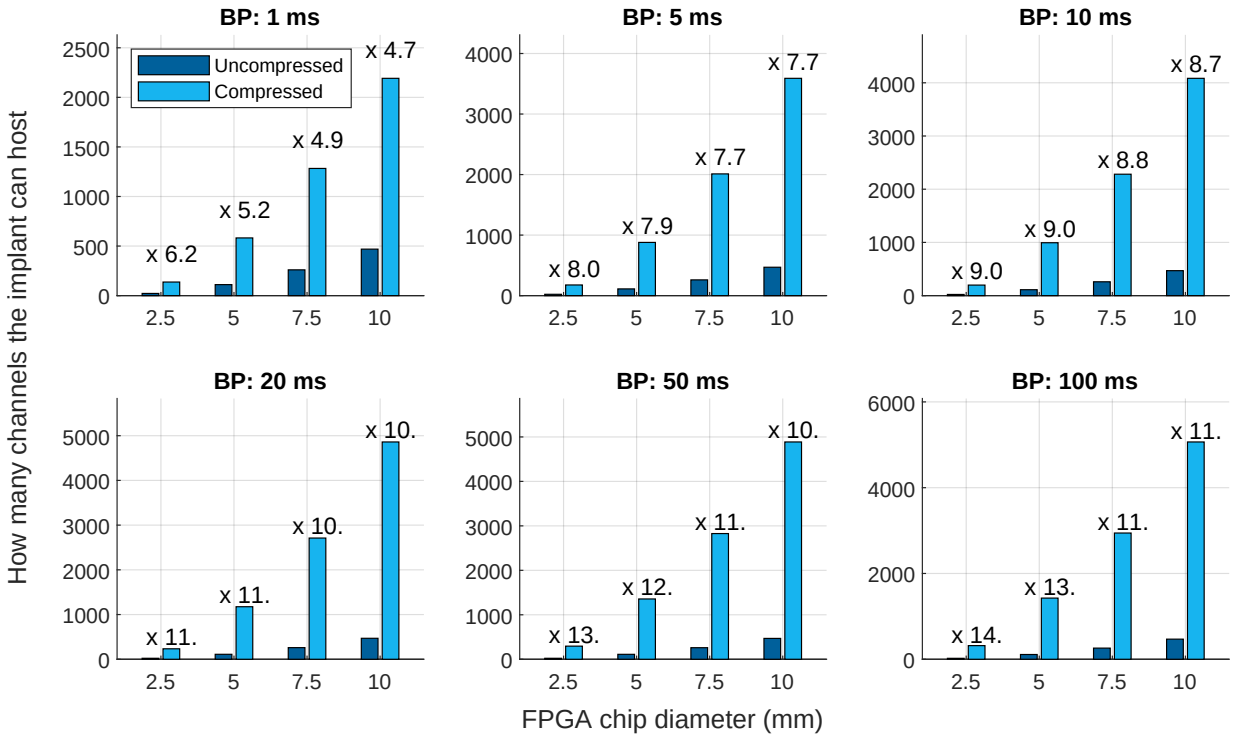

Figure 4: Number of MUA channels we can host on a Lattice ice40LP FPGA of different dimensions for the selected communication schemes at a BP $\in$ [1, 5, 10, 20, 50, 100] ms. This is compared to the number of channels that can be hosted given the standard uncompressed MUA representation at 1 ms BP, $S = 2$, that has a 1000 kbps/channel BR. The required power for any front-end ADC and pre-amplifiers is ignored. The total dynamic power budget is given by the FPGA dimensions multiplied by a heat flux limit of 10 mW/cm^2, minus the FPGA static power. For each BP and FPGA dimensions, the ratio of how many more channels can be fitted on-implant with compression is also given.

4.4 Sample Histograms for Mapping

The use of sample histograms to add adaptivity into the SH encodings, as in [13], was also investigated. Also to simplify the manuscript, detailed discussion and the BR results are given in the Supplemental Material. It was found that sample histograms did improve the BRs, but that their hardware and processing power costs, while not very large, were noticeable. It was deemed that, for a 20 nJ/bit communication energy, the use of sample histograms was not justified for the reduction in communication power. However, with larger communication energies, the use of sample histograms may be warranted to reduce the BR and so the communication power, which may be the dominating factor.

5 Conclusion

In this work, we looked at various algorithms for compressing MUA data for WI-BMIs. We made significant use of Static Huffman encoders because of their hardware efficiency and good compression performance. We found that they performed exceptionally well, and that the MUA signal could be compressed to varying degrees depending on BP. For example, at the standard BP of 1 ms, roughly 5-6 times more channels can be fitted onto the same mm-scale implant merely because of the addition of the delta-asynchronous compression scheme proposed in this work. MUA is already the most popular neural signal for WI-BMIs [3–5, 24]. The results from this work suggest that the MUA signal can be made even more attractive, given the possibility for hardware-efficient, ultra-low power compression.

References

[1] C. Pandarinath, P. Nuyujukian, C. H. Blabe, B. L. Sorice, J. Saab, F. R. Willett, L. R. Hochberg, K. V. Shenoy, and J. M. Henderson, "High performance communication by people with paralysis using an intracortical brain-computer interface," *Elife*, vol. 6, p. e18554, 2017.

[2] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "High-performance brain-to-text communication via handwriting," *Nature*, vol. 593, no. 7858, pp. 249–254, 2021.

[3] A. B. Rapeaux and T. G. Constandinou, "Implantable brain machine interfaces: first-in-human studies, technology challenges and trends," *Current Opinion in Biotechnology*, vol. 72, pp. 102–111, 2021, tissue, Cell and Pathway Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095816692100183X

[4] E. Musk *et al.*, "An integrated brain-machine interface platform with thousands of channels," *Journal of medical Internet research*, vol. 21, no. 10, p. e16194, 2019.

[5] N. Even-Chen, D. G. Muratore, S. D. Stavisky, L. R. Hochberg, J. M. Henderson, B. Murmann, and K. V. Shenoy, "Power-saving design opportunities for wireless intracortical brain–computer interfaces," *Nature Biomedical Engineering*, pp. 1–13, 2020.

[6] S. Todorova, P. Sadtler, A. Batista, S. Chase, and V. Ventura, "To sort or not to sort: the impact of spike-sorting on neural decoding performance," *Journal of neural engineering*, vol. 11, no. 5, p. 056005, 2014.

[7] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning," *Journal of Neural Engineering*, vol. 18, no. 2, p. 026011, 2021.

[8] E. M. Trautmann, S. D. Stavisky, S. Lahiri, K. C. Ames, M. T. Kaufman, D. J. O'Shea, S. Vyas, X. Sun, S. I. Ryu, S. Ganguli *et al.*, "Accurate estimation of neural population dynamics without spike sorting," *Neuron*, vol. 103, no. 2, pp. 292–308, 2019.

[9] G. W. Fraser, S. M. Chase, A. Whitford, and A. B. Schwartz, "Control of a brain–computer interface without spike sorting," *Journal of neural engineering*, vol. 6, no. 5, p. 055004, 2009.

[10] S. Luan, I. Williams, M. Maslik, Y. Liu, F. De Carvalho, A. Jackson, R. Q. Quiroga, and T. G. Constandinou, "Compact standalone platform for neural recording with real-time spike sorting and data logging," *Journal of neural engineering*, vol. 15, no. 4, p. 046014, 2018.

[11] L. Luan, J. T. Robinson, B. Aazhang, T. Chi, K. Yang, X. Li, H. Rathore, A. Singer, S. Yellapantula, Y. Fan *et al.*, "Recent advances in electrical neural interface engineering: minimal invasiveness, longevity, and scalability," *Neuron*, vol. 108, no. 2, pp. 302–321, 2020.

[12] Z. Zhang, O. W. Savolainen, and T. G. Constandinou, "Algorithm and hardware considerations for real-time neural signal on-implant processing," *Journal of Neural Engineering*, vol. 19, no. 1, p. 016029, 2022.

[13] O. W. Savolainen, Z. Zhang, P. Feng, and T. G. Constandinou, "Hardware-efficient compression of neural multi-unit activity using machine learning selected static huffman encoders," *bioRxiv*, 2022.

[14] P. D. Wolf and W. Reichert, "Thermal considerations for the design of an implanted cortical brain–machine interface (BMI)," *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment*, pp. 33–38, 2008.

[15] G. Gagnon-Turcotte, M. N. Khiarak, C. Ethier, Y. De Koninck, and B. Gosselin, "A 0.13-μm cmos soc for simultaneous multichannel optogenetics and neural recording," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3087–3100, 2018.

[16] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[17] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[18] O. W. Savolainen and T. G. Constandinou, "Lossless compression of intracortical extracellular neural recordings using non-adaptive huffman encoding," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2020, pp. 4318–4321.

[19] ——, "Investigating the effects of macaque primary motor cortex multi-unit activity binning period on behavioural decoding performance," *bioRxiv*, 2020.

[20] O. Savolainen, Z. Zhang, and T. Constandinou, "Code and results for 'comparison of data compression architectures for neural multi-unit activity signals in brain-machine interfaces'," https://github.com/Next-Generation-Neural-Interfaces/Comparison-of-ultra-low-power-hardware-efficient-MUA-compression-schemes, 2022.

[21] R. D. Flint, E. W. Lindberg, L. R. Jordan, L. E. Miller, and M. W. Slutzky, "Accurate decoding of reaching movements from field potentials in the absence of spikes," *Journal of neural engineering*, vol. 9, no. 4, p. 046006, 2012.

[22] J. E. O'Doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes, "Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology," May 2017. [Online]. Available: https://doi.org/10.5281/zenodo.788569

[23] T. Brochier, L. Zehl, Y. Hao, M. Duret, J. Sprenger, M. Denker, S. Grün, and A. Riehle, "Massively parallel recordings in macaque motor cortex during an instructed delayed reach-to-grasp task," *Scientific data*, vol. 5, no. 1, pp. 1–23, 2018.

[24] D. M. Brandman, T. Hosman, J. Saab, M. C. Burkhart, B. E. Shanahan, J. G. Ciancibello, A. A. Sarma, D. J. Milstein, C. E. Vargas-Irwin, B. Franco *et al.*, "Rapid calibration of an intracortical brain–computer interface for people with tetraplegia," *Journal of neural engineering*, vol. 15, no. 2, p. 026007, 2018.