Leveraging NVMs for Neural Interface Coverage

Muhammed Ugur
Yale University

Raghavendra Pradyumna Pothukuchi
Yale University

Abhishek Bhattacharjee
Yale University

Abstract

The data being read from invasive neural interfaces is increasing exponentially, stressing the safety and power constraints on these devices. There is a need to efficiently process and store this data on-device, especially as implants shift towards wireless deployments. Non-volatile memories (NVM) are an attractive option due to their persistence and energy efficiency. We argue that better NVM integration is needed going forward and that it will enable more flexible forms of deployment. However, doing this at low power with low latency and high capacity is critical. These are at odds with one another so new architectures and approaches are needed.

1 Introduction

Neural interfaces are an emerging treatment modality to help remove the burden of neurological and psychiatric disorders on patients with no remaining clinical options. They record neural activity and provide treatment through electrical stimulation. The most effective devices are invasive, recording higher fidelity signals and enabling better treatment with the requirement that they consume low power for safety. However, the amount of data being recorded is increasing exponentially, with the eventual goal of reading Tbps to record the billions of neurons in the brain.

Figure 1 shows an estimate for the data volume using historical and extrapolated data on simultaneously recorded neurons [8]. The data is fit using a double exponential model and attributes a sampling rate of 30000 Hz and 16-bit resolution to each neuron. Power consumption is estimated on the right axis for wirelessly transmitting data rates using a 200 pJ per bit radio. Given the strict milliwatt power constraints outlined by the FDA, transmitting all of this data wirelessly is unsafe. This motivates the need for on-device compute to either compress the data or perform treatment without having to go over the network.

On-device compute is realized in the form of accelerators for known neural interface kernels. These kernels are typically some variation of signal processing or machine learning algorithms. However, as data rates grow, power consumption increases within these accelerators. This is because they require more local memory, or SRAM, to process the independent streams of data, also known as channels. More efficient memory technologies are an option to alleviate this pressure.

A natural option is to integrate non-volatile memories or NVMs. They provide better density-per-watt compared to SRAM and scale better for exponentially-increasing data rates. Persistency is also a valuable feature. Existing devices already leverage non-volatility to store histories of recording or pre-defined templates for better treatment and diagnostics [1, 7]. As neural interfaces become more mobile and wireless, persistency will become necessary to stay within safe power constraints and deal with network disconnectivity. However, integrating NVM technologies is in its infancy and requires careful consideration to best use their features.

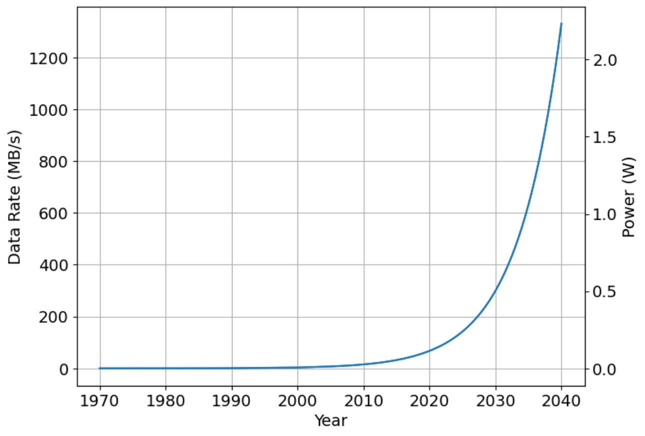
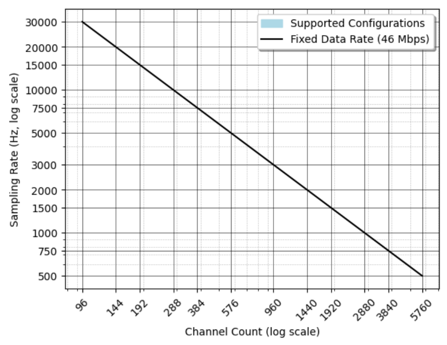


Figure 1. Exponential growth of data rates for electrophysiology-based neural recording with estimated power consumption of wireless transmission.

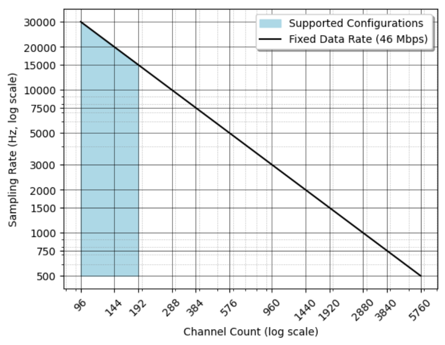
We have previously argued that re-thinking on-device compute to be swapping-centric by leveraging on-device storage in the form of NVMs will improve neural interface coverage [9]. However, doing this naively is inefficient and does not properly utilize the NVM to its highest potential. In this work, we expand our analysis and characterize how channel and sampling rate coverage varies depending on resource trade-offs. We also highlight important considerations when integrating NVMs such as their rich design space, reliability issues, and complexity.

2 Background

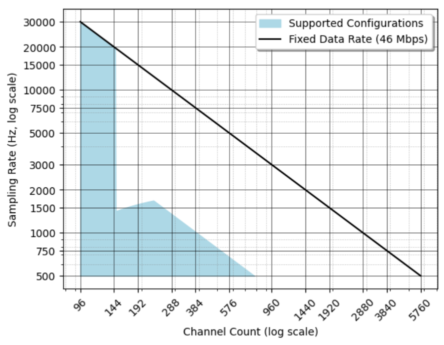
Both on-device compute and storage are becoming common for neural interfaces given the aforementioned tension between growing data rates and power constraints. Our previous work has looked into both integrating on-device compute and storage in a safe and effective manner [5, 7]. On-device storage is useful for storing longer histories of data, performing partial computations, and accessing pre-defined templates [7]. The NeuroPace RNS is a FDA-approved device used to treat refractory epilepsy which has around 1 MB of storage to store histories of neural activity [1]. Although this is useful, it is only a start and is far short from the Gbps that will eventually be read from the brain.



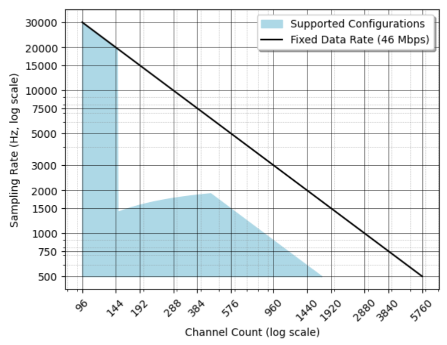
(a) Initial configuration support from [7].



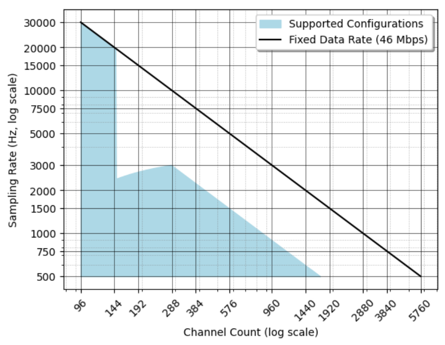
(b) Additional SRAM.



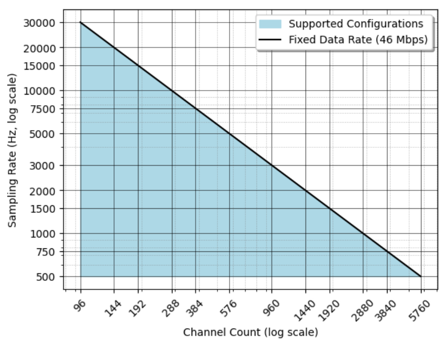
(c) Naive swapping for BBF.



(d) Additional NVM bandwidth.



(e) Lower NVM read/write latency.



(f) Ideal configuration support.

Figure 2. Shows the feasibility of different channel/sampling-rate configurations using a naive swapping approach for a signal processing accelerator (BBF). The different subfigures highlight how on-chip resources impact the coverage of the approach.

Given the infancy of on-device storage for invasive neural interfaces, there are still many open questions as to how they can be best integrated. In this work, we show that storage can be leveraged for more flexible deployments of neural interfaces. By flexible deployment we mean an interface that can support many different channel and sampling rate configurations from the recording probes, including channel counts much higher than those currently available today.

3 NVM Integration

By designing accelerators to be swapping-centric, i.e., intelligently moving data to and from the NVM, they can expand their coverage to more channel count and sampling rate configurations. Figure 2c shows the support of these configurations for the Butterworth Bandpass Filter (BBF) using a naive swapping approach. This approach is optimistic but it highlights how the latency and bandwidth of the NVM restrict coverage and how there is more room for improvement to maximize the potential of NVM integration.

3.1 NVM Modeling

This naive approach is modeled using characteristics from a Micron SLC NAND Flash chip [6]. This NVM is page-based and includes a chip, die, plane, block, and page hierarchy. Parallelism, in the form of parallel reads and writes, is allowed across chips, dies, and planes. For I/O, we include the cost of page movement across a shared bus and the latency of reads/writes based on the NVM. Read and write bandwidth is determined by bus characteristics and read/write latency.

3.2 Naive Swapping

For naive swapping, we first start out with an accelerator that supports only 96 channels at 30000 Hz (Figure 2a), which is based off of the system from [7]. One method for improving coverage is to add more SRAM (Figure 2b). However, this eventually hits a cap due to power consumption and is not preferable as we scale in the number of channels.

Instead, we design a naive approach which reads and writes the state of the filter for each channel. The idea is to first read the state of the filter from the NVM for every new sample, then compute the output of the filter. Then, update the state of the filter and write that state to the NVM. These I/Os are needed because there is not enough SRAM for the state of all the channels. This approach additionally considers buffering the data before a write and performing parallel writes. It also uses extra SRAM from the storage controller, giving immediate coverage for lower channel counts. What this technique enables is the expansion towards higher channel count systems at potentially lower sampling rates.

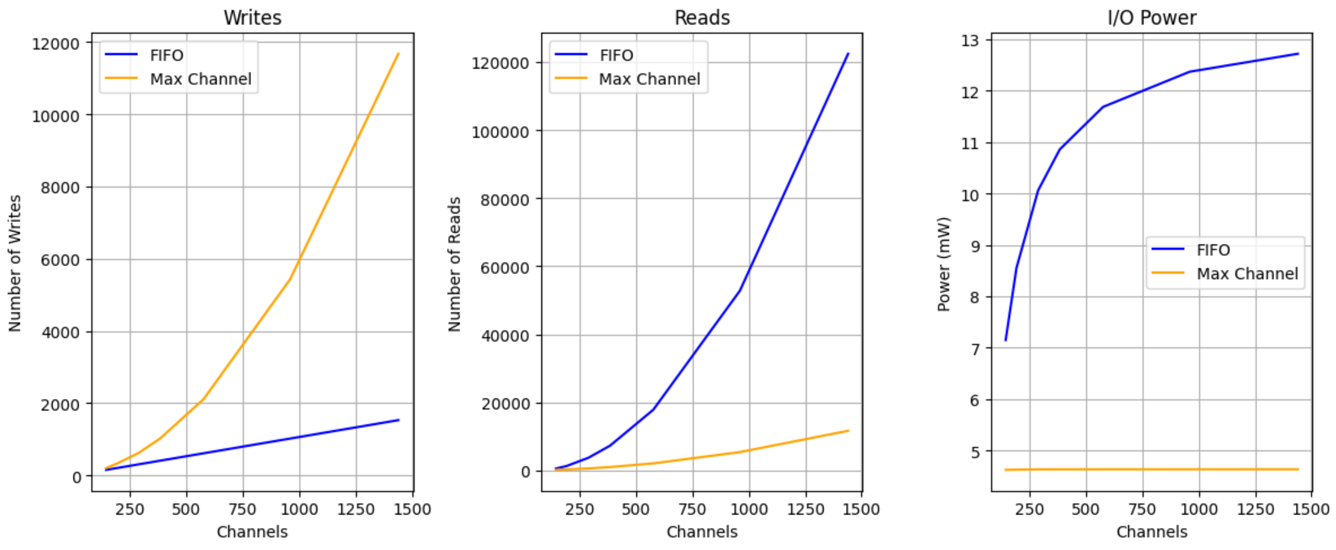


Figure 3. Showcases two swapping approaches for the Fast Fourier Transform (FFT) as channel counts increase. Swapping approaches can be read or write-centric, and depending on the relationship between read/write power for a given NVM, the power consumption can vary substantially. Here the NVM's instantenous read power is much higher than the write power.

This is still practical since patients require personalized care for their diagnoses and treatments. Lower sampling rates may be tolerable or preferred if more channels are allowed and vice versa.

3.3 System Trade-offs

Figure 2d shows how increasing the bandwidth three-fold impacts the coverage of the naive swapping approach. NVM bandwidth saturates quickly since the naive swapping generates a read and write for each channel within a short window of time. Bandwidth allows us to cover more channel counts and also sampling rates and pushes coverage towards the diagonal line, which represents the data rate the system was originally designed for. If less I/Os are performed, e.g., by batching the state of the filter across many channels, then coverage would expand further.

Figure 2e shows how a lower read and write latency to the NVM increases coverage based on the previous increase in bandwidth. Here we are able cover more under the bandwidth constraint, specifically lower channel counts with higher sampling rates. This is because lower channel counts are processed quickly, so a read and write must finish sooner to process the next sample. When the latency of the NVM is halved, then more of these configurations can be supported.

Finally, Figure 2f shows the ideal case where we are able to support all channel and sampling-rate configurations for the fixed data rate. This fixed data rate scenario represents the case where on-device compute is clocked at the same frequency even as channel counts increase. Ideally, this diagonal is pushed further to the right, which can be done by improving the efficiency of on-device compute. This will allow the accelerators to be clocked much higher, and therefore, process higher data rates.

3.4 Read/Write Power

Figure 3 showcases two different swapping approaches, called FIFO and Max Channel, for the Fast Fourier Transform (FFT). FIFO is an approach which receives samples from each channel in its natural order, and when SRAM is full, it simply writes this first-in first-out layout to the NVM on a page-by-page basis. Once all data is received, the approach will then read the necessary pages and samples to compute each channel's FFT. This approach incurs low writes but a large number of reads.

Max Channel is a different swapping method which emphasizes more writes for less reads. The key idea here is to evict the channel with the most samples in SRAM to its own page once SRAM is at capacity. This provides spatial locality and reduces the amount of reads later on when computing the FFT of each channel. The first two graphs in Figure 3 showcase the trade-off in reads and writes between both FIFO and Max Channel.

The graph furthest on the right estimates the power consumption in milliwatts of these approaches for a selected NVM. The energy estimates for this NVM were collected using NVSim [4]. The key observation here is that with NVSim, the instantenous power of a read page is much higher than writing a page. This means that for a swapping approach that is more read-centric, like FIFO, the overall power consumption will be higher.

For FIFO, the increasing trend in power consumption as channels increase is because of the increasing ratio between reads and writes. FIFO does not scale well with higher channel counts because spatial locality suffers under this approach. Max Channel on the other hand has a stable read/write ratio as channels increase. The overall power consumption of Max Channel is therefore less because the power of a write for this selected NVM is much lower than a read. The downside with Max Channel is that writes have high latency and implementing this approach in real-time may not be feasible. The insight with this example is that device characteristics, namely read and write power, will impact the best swapping approach. Improving swapping approaches, like the naive approach in Figure 2c, will need to be device-aware going forward.

3.5 End-to-End Applications

The approaches presented so far are for a single accelerator. However, there are typically many accelerators working together to realize an end-to-end pipeline or application [5, 7]. Applying these swapping techniques across many accelerators at once will be challenging but necessary to realize full NVM integration. To do so will require understanding how the raw signal data changes throughout an end-to-end pipeline. If data is filtered, for example, then swapping may not be necessary or be relatively cheap to implement. Many of the accelerators which generate features, such as BBF or FFT, create the most pressure on the NVM due to their proximity to the raw signal data. Identifying overlap between data will be critical to minimize I/Os to the NVM.

3.6 Reliability

Integrating NVMs not only requires carefully understanding system resources and power, but also device reliability of the underlying memory technology. Endurance, for example, is a critical issue that needs to be addressed before integrating an NVM [3]. NVMs may be susceptible to wear-out, where each block of data has a limited amount of writes before being unreliable. Within the context of neural interfaces, improperly handling wear-out or increasing the amount of writes will chip away at the lifetime of the implant, possibly leading to risky and costly replacement surgery.

There are other reliability considerations as well. This includes handling read and write disturbances, using error-correcting codes, handling garbage collection, and more [3]. Depending on the requirements set forth by the clinicians, these need to be addressed to avoid data loss and maintain accuracy. However, supporting reliability comes with overhead which will take away resources that would have otherwise been allocated to better performance or higher data rates.

3.7 Technology

The analysis so far has focused on NAND Flash due to its accessibility and practicality. Other memory technologies could also be characterized and integrated such as Resistive RAM (RRAM), Ferroelectric RAM (FRAM), Magnetoresistive RAM (MRAM), Phase-Change Memory (PCM), and Spin-Transfer Torque RAM (STT-RAM) [2, 4]. These technologies would impact the energy/power consumption, latency, capacity, and addressability of the swapping approaches. A systematic study of the trade-offs is necessary to maximize neural interface coverage.

4 Conclusion

There are many factors that go into integrating NVMs for neural interfaces. This includes the choice of technology, its performance characteristics, handling reliability and wear-out, and the relationship with on-device compute. This work presented these trade-offs and introduced a naive approach for leveraging NVMs for better channel and sampling-rate coverage. However, more can be done to expand coverage due to the algorithms for on-device compute being known and predictable. Operating in this regime of specialization allows us to invoke theory to maximize our system coverage. Given the clinical objectives of neural interfaces, pursuing this will be imperative for architects to ensure the best possible treatment and quality-of-life to patients.

References

[1] [n. d.]. RNS System Physician Manual. https://www.neuropace.com/wp-content/uploads/2021/02/neuropace-rns-system-manual-320.pdf. Accessed: 2023-11-03.

[2] Satyajaswanth Badri, Mukesh Saini, and Neeraj Goel. 2023. An Efficient NVM-Based Architecture for Intermittent Computing Under Energy Constraints. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 31, 6 (June 2023), 725–737. https://doi.org/10.1109/tvlsi.2023.3266555

[3] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. 2017. Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery. https://doi.org/10.48550/ARXIV.1711.11427

[4] Xiangyu Dong, Cong Xu, Yuan Xie, and N. P. Jouppi. 2012. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 7 (July 2012), 994–1007. https://doi.org/10.1109/tcad.2012.2185930

[5] 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)*. IEEE, 391–404.

[6] Inc. Micron Technology. [n. d.]. MT29F128G08AKCABH2-10. https://www.micron.com/products/nand-flash/slc-nand/part-catalog/mt29f128g08akcabh2-10. Retrieved December 22, 2023.

[7] Karthik Sriram, Raghavendra Pradyumna Pothukuchi, Michał Gerasimiuk, Muhammed Ugur, Oliver Ye, Rajit Manohar, Anurag Khandelwal, and Abhishek Bhattacharjee. 2023. SCALO: An Accelerator-Rich Distributed System for Scalable Brain-Computer Interfacing. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23)*. ACM. https://doi.org/10.1145/3579371.3589107

[8] Ian H Stevenson and Konrad P Kording. 2011. How advances in neural recording affect data analysis. *Nature Neuroscience* 14, 2 (Jan. 2011), 139–142. https://doi.org/10.1038/nn.2731

[9] Muhammed Ugur, Raghavendra Pradyumna Pothukuchi, and Abhishek Bhattacharjee. 2024. Swapping-Centric Neural Recording Systems. In *The 15th Annual Non-Volatile Memories Workshop.*