Using Branch Predictors to Predict Brain Activity in Brain-Machine Implants

Abhishek Bhattacharjee
Department of Computer Science, Rutgers University
Princeton Neuroscience Institute*, Princeton University
abhib@cs.rutgers.edu

ABSTRACT

A key problem with implantable brain-machine interfaces is that they need extreme energy efficiency. One way of lowering energy consumption is to use the low power modes available on the processors embedded in these devices. We present a technique to predict when neuronal activity of interest is likely to occur so that the processor can run at nominal operating frequency at those times, and be placed in low power modes otherwise. To achieve this, we discover that branch predictors can also predict brain activity. We perform brain surgeries on awake and anesthetized mice, and evaluate the ability of several branch predictors to predict neuronal activity in the cerebellum. We find that perceptron branch predictors can predict cerebellar activity with accuracies as high as 85%. Consequently, we co-opt branch predictors to dictate when to transition between low power and normal operating modes, saving as much as 59% of processor energy.

CCS CONCEPTS

• **Computer systems organization** → **Special purpose systems**; **Embedded hardware**; **Real-time system architecture**; • **Hardware** → **Bio-embedded electronics**; **Neural systems**;

KEYWORDS

Brain-machine interfaces, neuroprostheses, embedded processors, branch predictors, perceptrons, power, energy.

ACM Reference format:
Abhishek Bhattacharjee. 2017. Using Branch Predictors to Predict Brain Activity in Brain-Machine Implants . In *Proceedings of MICRO-50, Cambridge, MA, USA, October 14–18, 2017,* 14 pages.
https://doi.org/10.1145/3123939.3123943

1 INTRODUCTION

Recent advances in invasive/non-invasive brain monitoring technologies and neuroprostheses have begun shedding light on brain function. Brain-machine interfaces for persons afflicted by epilepsy, spinal cord injuries, motor neuron diseases, and locked-in syndrome are undergoing rapid innovation [18, 20, 38, 46, 47, 64, 72]. This is partly because the technologies used to probe and record neuronal activity in vivo are fast improving – we can currently monitor the activity of hundreds of neurons simultaneously, and this number is doubling approximately every seven years [63]. This means that scientists can now study large-scale neuronal dynamics and draw connections between their biology and higher-level cognition.

Consequently, scientists are integrating embedded processors on neuroprostheses to achieve more sophisticated computation than what was previously possible with the micro-controllers and analog hardware traditionally used on these devices [6, 20, 23, 38, 45, 50, 64, 72]. For example, designers are beginning to use embedded processors for sub-millisecond spike detection and sorting to apply stimuli to the brain whenever a specific neuron fires [52, 72]. Similarly, new-brain machine interface designs use embedded processors rather than bulky and inconvenient wired connections to large desktops [7, 20, 44, 54].

These processors face an obstacle – they need to be energy efficient. Consider the cerebellum, which resides in the hindbrain of all vertebrates. Recent studies use invasive brain monitoring to record intracellular cerebellar neuronal activity [19, 53, 65]. Invasive brain-machine implants cannot typically exceed stringent 50-300mW power budgets [6, 23, 45, 50, 64, 72]. This is because neural implants have small form factors and must, therefore, use the limited lifetimes of their small batteries judiciously [6, 23, 45, 50, 64, 72]. Equally importantly, stretching out battery lifetimes can reduce how often invasive surgeries for battery replacement and/or recharging are needed. Finally, power consumption must be kept low, as temperature increases in excess of 1-2 degrees celsius can damage brain tissue [39, 69, 70]. Unfortunately, the embedded processors used on implants can currently hamper energy efficiency in some cases, expending 30-40% of system energy [32, 64, 72].

A potential solution is to use the low power modes already available on these processors [15–17, 22, 30, 34]. Traditional energy management on server and mobile systems balance the energy savings of low power modes with performance degradation, by anticipating periods of time when applications do not need certain resources or can afford a slowdown [10, 15–17, 30, 35, 41–43]. Similar approaches are potentially applicable to brain implants. Since embedded processors on implants perform signal processing on neuronal spiking data, they could theoretically be placed in low power mode in the absence of neuronal firing and be brought back to nominal operation before neuronal activity of interest. This presents the following question – how can we predict when future neuronal spiking is likely to occur, both accurately and efficiently?

*The author performed the experimental work in this paper at the Princeton Neuroscience Institute, where he is currently a 2017-2018 CV Starr Visiting Fellow.

MICRO-50, October 14–18, 2017, Cambridge, MA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4952-9/17/10…$15.00
https://doi.org/10.1145/3123939.3123943

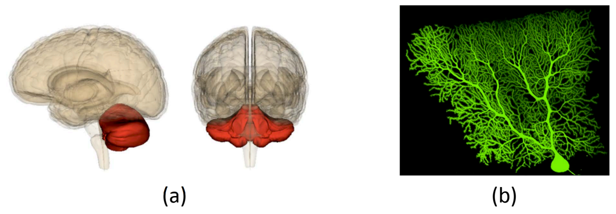

Figure 1: (a) The cerebellum, shown in red, is located behind the top of the brain stem and has two hemispheres [1]; (b) a major cerebellar neuron is the Purkinje neuron, imaged here from a mouse brain [9].

In response, we note that architects have historically implemented performance-critical microarchitectural structures to predict future program behavior. One such structure, the branch predictor, is a natural fit for neuronal prediction too. Branch predictors assess whether a branch is likely to be taken or not, and as it turns out, map well to the question of whether a neuron fires or not at an instant in time. We study many branch predictors and discover that the perceptron predictor [5, 24–27] accurately predicts future cerebellar neuronal activity. We co-opt the perceptron predictor to not only predict program behavior but to also manage the low power modes of a cerebellar implant. Our contributions are:

① We evaluate well-known dynamic hardware branch predictors, including Smith predictors [60], gshare [40], two-level adaptive predictors [71], and the perceptron predictor [26]. We perform surgeries on awake and anesthetized mice to extract 26 minutes of neuronal spiking activity from their cerebella and find that perceptron branch predictors are particularly effective at predicting neuronal activity, with up to 85% accuracy. The success of the perceptron predictor can be attributed to the fact that it captures correlations within long histories of branches better than other approaches. This fits well with cerebellar neuronal activity, where groups of neurons also tend to have correlated activity [53, 65].

② We model a cerebellar monitoring implant and use the embedded processor's branch predictor to guide energy management. We place the processor in idle low power mode but leave (part of) the predictor on. When the predictor anticipates interesting future neuronal activity, it returns the processor to nominal operation. We use architectural, RTL, and circuit modeling, and find that this approach saves up to 59% processor energy.

A theme of this work is to ask – since machine learning techniques inspired by the brain have been distilled into hardware predictors (e.g., like the perceptron branch predictor), can we now close the loop and use such predictors to anticipate brain activity and manage resources on neuroprostheses? Our work is a first step in answering this question. Ultimately, this approach can guide not only management of energy but also other scarce resources.

2 BACKGROUND

The Cerebellum: The cerebellum affects motor control, language, attention, and regulates fear and pleasure responses [19, 53, 65, 67]. It receives input from the sensory systems of the spinal cord and from other parts of the brain, integrating them to fine-tune motor activity. Cerebellar damage leads to movement, equilibrium, and motor learning disorders. Cerebellar damage may also play a role in hallucination and psychosis [11, 29, 56]. Figure 1 shows the location of the cerebellum in the human brain and an in vivo image of one of its major neuron types, the Purkinje neuron. Our goal is to enable energy-efficient recording of Purkinje activity.

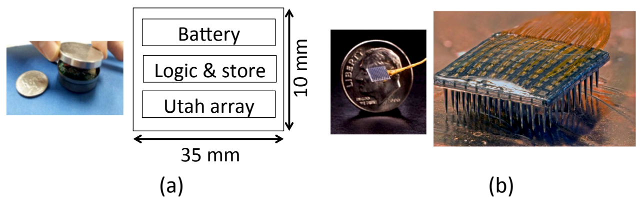

Figure 2: (a) Block diagram of a cerebellar implant (dimensions not drawn to scale) and compared against a coin [64]; (b) the Utah array is used to collect intracellular Purkinje recordings [2, 3].

Cerebellar Monitoring: Figure 2 shows a typical cerebellar implant [64, 72]. Implants are small and placed in containers embedded via a hole excised in the skull, from where they probe brain tissue. Figure 2 shows the following:

① Microelectrode array: In vivo neuronal activity is picked up using microelectrode arrays, which have improved rapidly in recent years [63]. Many implants, including our target system, use Utah arrays made up of several tens of conductive silicon needles that capture intracellular recordings [31, 51, 66]. Utah arrays are widely used because of their high signal fidelity, robustness, and relative ease of use.

② Logic/storage: Neuronal activity recorded by the Utah array is boosted by analog amplifier arrays connected to analog-to-digital converters (ADCs). 16-channel ADCs produce good signal integrity without excessive energy usage [6, 64]. ADCs route amplified data to locations in LPDDR DRAM. Flash memory is used to store neuronal data [45]. Since gigabytes of neuronal activity data can be generated in just tens of minutes of recording, most implants use a wireless communication link (typically a GHz RF link) to transmit data to a desktop system with sufficient storage for all the data being recorded. Embedded processors (e.g., energy-efficient ARM Cortex M cores) are integrated on these implants [6, 45, 64, 72]. We focus on an implant with an embedded processor with similar microarchitecture to the Cortex M7 (see Section 7). These processors run at 200-300 MHz, but maintain two low-power modes to turn either the processor clock off (to roughly halve processor power consumption) or turn off DRAM and flash too (to lower system power consumption by an order of magnitude) [14].

③ Battery: Designers often use 3.7 Volt batteries to power implants and target lifetimes of days to weeks for mouse studies. Target timescales increase to several weeks for primate studies. Longer lifetimes are better, reducing the need for surgeries to replace batteries [20]. Wireless charging can also reduce the need for surgeries; nevertheless, energy efficiency remains important because implants must not raise temperature beyond 1-2 degrees celsius to prevent tissue damage [39, 69, 70]. As a result, designers aim to run implants with power budgets of 50-100mW, occasionally permitting short timescales of 300mW budgets [6, 45, 64, 72].

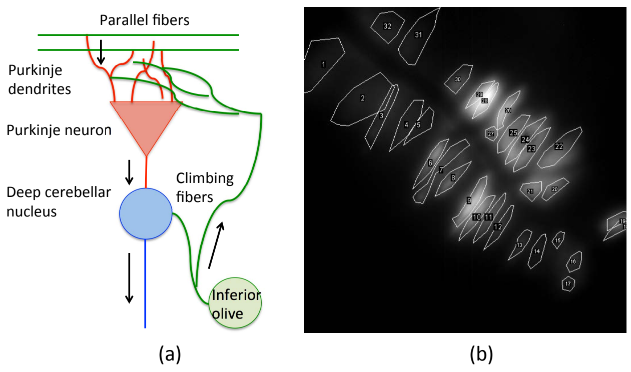

Figure 3: (a) Purkinje neurons are activated by the inferior olive and parallel fibers; and (b) an example of synchronized activity (we surgically collect this image from the cerebellum of a mouse, with Purkinje neurons outlined).

3 MOTIVATION

Embedded processors can consume 30-40% of system-wide energy in modern brain-machine implants, with components like DRAM, flash, and analog circuits consuming the rest [64]. We attack this bottleneck using low power modes. To do this, we answer several important questions:

What low power modes do we use? We study signal processing workloads that perform useful computation when there is neuronal activity of interest. Correct operation requires memory and architectural to be preserved during low power mode. Deep low power modes that lose state are hence infeasible. On ARM Cortex processors, which use the low power modes detailed in Section 2, we use modes that turn off the processor and caches but not DRAM or flash. In the future, as Cortex M7 processors adopt stateful low power modes for DRAM and flash memory [15, 17], we anticipate using them too.

When can we use low power modes? To use low power modes until neuronal activity of interest, we must define the notion of interesting activity. Since neural implants are used for many tasks, this definition can vary. Our study monitors Purkinje neurons in the cerebellum. Purkinje firing can be separated into two classes – unsynchronized and synchronized firing. To understand their differences, consider Figure 3(a), which shows the cellular anatomy of a Purkinje neuron.

Cerebellar Purkinje neurons are driven by two inputs. The first is a set of parallel fibers which relay activity from other parts of the cerebellum. Parallel fibers are connected to Purkinje neurons using the spindly outgrowths of the neurons, i.e., their dendrites. The second input is the inferior olivary nucleus, which provides information about sensorimotor stimulation [73]. Olivary nuclei are connected to climbing fibers, which feed Purkinje dendrites.

When either the parallel fibers or the inferior olive fire, spikes are activated on the Purkinje neuron. These spikes drive the deep cerebellar nucleus, influencing motor control and longer-term cerebellar plasticity [53]. The exact nature of Purkinje activity depends on the input that triggered the Purkinje neuron. Purkinje spikes due to parallel fibers occur at 17-150 Hz, while those prompted by the inferior olivary nuclei occur at 1-5 Hz [53].

Neuroscientists are studying many aspects of Purkinje spiking, but one that is important is that of synchronized spiking [29, 53, 65, 67]. While single Purkinje neurons usually fire seemingly in isolation, occasionally clusters of Purkinje neurons fire close together in time. Such synchronized firing usually occurs when neighboring olivary nuclei are activated in unison. Figure 3(b) shows imaging data we collect from an anesthetized mouse, where Purkinje neurons have been outlined. The flashing neurons represent firing while those in black represent quiescence. In the time slice shown, several Purkinje neurons fire synchronously.

Given their importance, synchronized firing is our focus. We enable energy-efficiency by using low power modes when Purkinje synchronization is absent, and permit nominal operation when synchronized activity occurs. In so doing, we sustain longer battery lifetime and collect longer and more thorough neuronal recording data for brain mapping studies.

How many Purkinje neurons must fire to be considered synchronized? This depends on what neuroscientists are studying. Some studies consider synchronization to require at least two neurons to fire, while others require four, eight, or more neurons to fire [53]. Naturally, requiring more neurons to fire makes synchronization rarer, affording more opportunity to save energy. Our goal is to save energy by exploiting any opportunity afforded by the nature of how scientists define synchronization. Section 8 shows how energy savings vary with different synchronization thresholds.

Why do we need neuronal activity prediction? One may initially expect to achieve energy efficiency by placing the processor in sleep mode until the microelectrode array captures synchronized Purkinje activity. At this point, the processor could be transitioned to nominal operating frequency. The problem with this approach is that scientists are curious not just about synchronized activity, but also about milliseconds of individual neuronal activity leading up to synchronized firing [53, 65]. Hence, it is better to anticipate or predict neuronal synchronization ahead of time so that events leading up to it are also recorded as often as possible.

How much energy can we potentially save with neuronal prediction? Having qualitatively discussed the benefits of neuronal activity prediction, we now quantify these benefits. We model a baseline with a 300 MHz ARM Cortex M7 processor, and run four neuronal processing workloads (see Section 7 for details). The workloads read the data picked up by the Utah array and process it to assess whether it represents synchronized activity. When the workloads identify synchronized activity, they perform signal processing on all neuronal activity (synchronized or unsynchronized) in the next 500ms. They then again read neuronal data to assess when the next synchronized event occurs. This baseline does not use the Cortex M7's idle low power modes because the workloads either continuously profile the neuronal data to detect synchronized activity or process neuronal spiking during and right after synchronization. Without the ability to predict Purkinje synchronization, the processor cannot know when it is safe to pause execution and use idle low power modes.

We contrast the baseline against an ideal – and hence unrealizable – oracle neuronal activity predictor that knows the future,

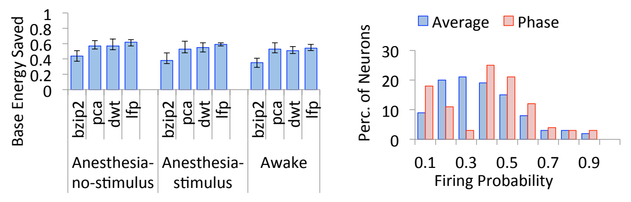

Figure 4: (Left) energy savings from perfect synchronization prediction, assuming four neurons must fire for synchronization. Anesthesia-no-stimulus **and** Anesthesia-stimulus **represent cases where the mouse is under anesthesia without and with stimulus, while** Awake **corresponds to non-anesthetized mice; and (right) neuronal spiking probability of an awake mouse on average** (Average) **and for a 5s phase** (Phase)**.**

and imposes no area, performance, or energy overheads. This oracle predictor views time in epochs, and is invoked at the end of each epoch to predict whether synchronized Purkinje activity is to occur in the next epoch. Based on the timescales that Purkinje spikes are sustained [53], we assume 10ms epochs. For the purposes of this discussion, we assume that four firing neurons constitute synchronization for now[1]. The processor is suspended in sleep state until the oracle predictor anticipates synchronization. In response, the processor transitions to nominal operation, capturing both the 10ms leadup time to synchronized activity, and the following 500ms of activity. We also model transition times between power modes. Since these take tens of microseconds on Cortex M processors, they have little impact on millisecond-ranging epochs [62]. The graph on the left in Figure 4 quantifies the results for 26 minutes of neuronal activity, split into three classes:

① Anesthesia without stimulus: We anesthetize mice and extract seven 2-minute traces of 32 Purkinje neurons. Anesthetized mice exhibit little movement aside from occasional spontaneous twitching of the limbs, whiskers, and tail.

② Anesthesia with stimulus: To study the effect of controlled sensory stimuli on Purkinje neurons, like past work [53], we apply 20-40 psi air puffs every 1s to the whiskers of the anesthetized mice. We collect three traces of Purkinje activity, each 2 minutes long. The sensorimotor stimulation from the air puffs increases Purkinje synchronization [53].

③ Awake: We collect three 2-minute neuronal traces from an awake free-roaming mouse. The rate of synchronized Purkinje firing varies depending on how the mouse moves.

Figure 4 shows that all benchmarks stand to enjoy significant energy benefits in every single case. We separate energy benefits into average numbers for each of the traces in ①-③, also showing the minimum and maximum values with error bars. With an ideal Purkinje synchronization predictor, energy savings can span 29-65% of total processor energy. Naturally, as Purkinje synchronizations become more frequent (either because mice are stimulated with air puffs or are awake), energy benefits drop since the processor cannot be placed in sleep mode as often. Still, even in these cases, 63% of energy can be saved with ideal predictors.

[1] Section 8 shows the impact of varying the number of neurons that must fire to constitute a synchronized event.

Why do we use branch predictors? If synchronized firing were to occur with well-defined periodicity, prediction would be simple, but this is not the case [19, 53]. Consequently, we consider alternatives. We were intrigued by the prospect of using branch predictors for neuronal prediction as the binary nature of taken/not-taken branches maps naturally to the notion of Purkinje neuron firing/quiescence. Additionally, modern branch predictors rely on local history and inter-branch correlations. Fundamentally, neuronal prediction relies on the same features. For example, neuroscientists have established that different Purkinje neurons exhibit different spiking likelihoods [53, 67]. We show, in the graph on the right in Figure 4, what percent of neuron samples in our awake mouse traces achieve firing probabilities of 10%, 20%, and so on. We separate average results from a particular 5s phase in the traces. While some neurons (8% on average) are biased towards quiescence and spike up to 10% of the time, as many as 40% of them can spike 10-30% of the time. A further 34% of them spike as often as 30-50% of the time. We also find that these spiking rates vary substantially based on what the mouse is doing. For example, the Phase data shows that in a particular 5s window, neuronal activity becomes more bimodal with more neurons being quiescent or active compared to the average. The complexity and variations of these spiking patterns mean that a simple history-based predictor which predicts that the next epoch would have the same spiking as the current epoch accurately predicts only 10-18% of the time. Therefore, as with branches, local history is helpful but does not alone yield high prediction accuracy. Fortunately, several studies show that Purkinje neurons, like branches, exhibit correlated spiking [53, 67]. Intuitively, branch prediction is a good candidate for neuronal prediction because it is based on a combination of local history and inter-branch/neuron correlation.

Is it feasible to use branch predictors on implants? Beyond their functional suitability for predicting neuronal activity, branch predictors are also a feasible choice because they are already being used on modern implants. This may be surprising as one may initially expect implants to use only simple micro-controllers, which do not need branch predictors. Early implant designs did just that, recording neuronal activity and relaying it wirelessly to a desktop for further processing. But implant designs are now changing rapidly. The number of neurons we can probe is doubling every seven years [63], so we are extracting the activity of several tens to hundreds of neurons today. The more neurons we probe from, the more sophisticated the desired processing. Processing must often be in real time, especially if the implant is expected to stimulate a neuron in response to neuronal activity. Designers therefore increasingly prefer on-implant processing, reducing wireless transmission to desktops [64]. Second, wireless transmission is a significant contributor to the total heat burden of the implant [69]. Consequently, emerging implants are integrating far more complex processors (e.g., ARM M4/M7 cores) to reduce wireless transmission. While many of these processors remain in-order, they routinely incorporate optimizations like pipeline forwarding, cache hierarchies, and branch predictors [38, 64].

To more quantitatively show the benefits of branch predictors on implants, consider processors like the ARM Cortex M7, which is being studied for neural implants today and which uses Smith predictors with two-bit saturating counters [60]. Since we need aggressive branch prediction to achieve the potential energy savings described in Figure 4, we perform an area-equivalent energy analysis of branch predictors for our baseline system without low power modes. We find that aggressive branch prediction can actually save energy – Smith predictors save 13% average energy compared to a case without branch prediction, by reducing workload runtime. Gshare and two-level adaptive predictors save a further 4% average energy by shortening runtime and reducing mispredictions/wrong-path execution. But more complex perceptron predictors increase average energy usage by 5% compared to Smith predictors. While this may seem to be a problem, our results in Section 7 show that perceptrons more accurately predict spiking; therefore, when we do use low power modes, perceptrons identify more opportunitiess to save energy than other predictors. Ultimately, this leads to more energy-efficient implants.

Sampling or software-based prediction techniques? It is reasonable to ask, at this point, whether software learning approaches that are more flexible and powerful than hardware branch predictors may be a better choice for neuronal activity prediction. Indeed, we believe that it may be possible to run learning algorithms on the implant to analyze neuronal behavior and more accurately predict future spiking activity. The downside of this approach is that valuable CPU/memory resources – and energy – must be consumed to enable this. Whether the potentially higher neuronal prediction accuracy of this approach ultimately leads to better energy savings than using hardware branch predictors warrants detailed study that we leave for future work.

One may also consider that using branch predictors for neuronal prediction has similarities with non-deterministic sampling. A natural question is – how would standard deterministic sampling techniques, which can be easier to implement, fare instead? We believe that the two approaches are orthogonal and likely complementary. Section 8 quantifies the potential of deterministic sampling, shedding light on what aspects of their operation may help neuronal prediction, and what aspects require deeper exploration. A key drawback with sampling is that it does not easily capture the lead-up activity to synchronization since one cannot tell, at the moment in time when a sample is taken, whether the activity corresponds to pre-synchronization firing.

4 IMPLEMENTATION

We use branch predictors for neuronal prediction as they have been implemented for decades, so vendors know how to design them efficiently and correctly. But there are other ways in which the branch predictor (and generally, learning-based hardware) may be co-opted for neuronal prediction. Though we drive our design with an implementation where the embedded processor's existing branch predictor predicts Purkinje spiking, we discuss other design options in Section 6.

4.1 Energy Management Strategies

Figure 5 shows how we manage energy. Since Purkinje activity is usually unsynchronized, the Cortex M7 is placed in idle low power mode, turning off processor and cache clocks but not DRAM. For this discussion, we cannibalize the branch predictor that exists in

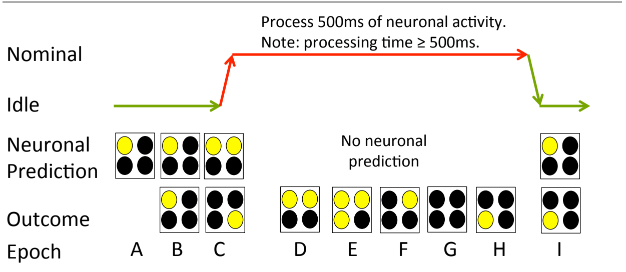

Figure 5: The processor is suspended in idle low power mode, but part of the branch predictor is kept on to predict Purkinje spiking. When it correctly predicts synchronized Purkinje firing, the processor goes to nominal operating frequency. In this example, synchronization occurs when at least two of the four neurons fire.

the embedded core to perform neuronal prediction. Our Idle state differs from traditional low power modes by keeping the branch predictor on. We implement a hardware FSM to guide neuronal prediction (see Section 4.2). Figure 5 shows Neuronal Predictions and actual Outcomes, split into Epochs of time labeled A-I (10ms in our studies). Our example monitors four neurons, shown in circles. Yellow and black circles represent firing and quiescence, respectively. In this example, synchronization occurs when at least two neurons fire.

In epoch A, the branch predictor is used for neuronal prediction and predicts that only a single Purkinje neuron will fire in the next epoch, B. Consequently, the processor continues in Idle. The prediction is correct as it matches the Outcome in B[2]. Simultaneously in B, the predictor predicts that only one neuron will fire in C. This turns out to be correct again – although the exact neuron that fires does not match the prediction, a concept that we will revisit shortly – and the processor continues in Idle. However, in C, the predictor anticipates a synchronization event between the top two Purkinje neurons. Consequently, the processor is transitioned into Nominal operating mode. Since transition times on the Cortex M7 are orders of magnitude smaller than 10ms epoch times [62], our prediction enables us to awaken the processor sufficiently early to process not only synchronization activity but also activity leading up to it. Once in nominal operation, the processor analyzes 500ms of Purkinje neuron activity, which can consist of synchronized and unsynchronized spiking, as shown in D-E and F-H respectively. During this time, the branch predictor returns to predicting branches and not neuronal activity. Note that the time taken to analyze 500ms of neuronal activity can exceed 500ms. Finally, the processor again transitions to Idle, with the branch predictor returning to brain activity prediction. Overall, there are four possible combinations of neuronal prediction and outcomes:

① Correctly predicted non-synchronization: This is the desirable combination, as it allows the processor to idle as long and often as possible.

② Correctly predicted synchronization: We want most synchronizations to be correctly predicted, enabling capture of both the activity before synchronization, as well as 500ms of neuronal activity during and after it.

[2]Section 4.2 explains that neuronal activity outcomes are provided by the Utah array and ADCs, which place data in DRAM.

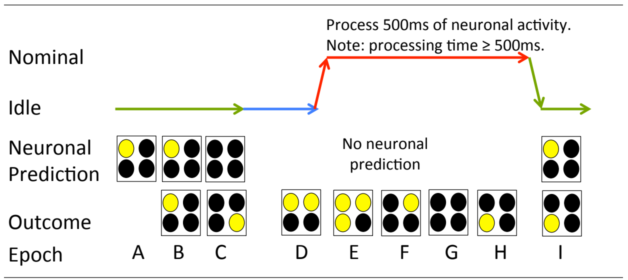

Figure 6: The branch predictor can mispredict neuronal activity. In this figure, it misses upcoming Purkinje synchronization, so the processor does not record 10ms events leading up to synchronization (in blue), though it is woken up when the misprediction is identified.

③ Incorrectly predicted non-synchronization: Branch predictors can mispredict. Figure 6 shows that in epoch C, the predictor expects no Purkinje neurons to fire in epoch D. This prediction is incorrect, as the top two Purkinje neurons do fire in D. We mitigate the damage caused by this by transitioning to Nominal operating mode as soon as we detect the misprediction. Unfortunately, the implant still misses the opportunity to monitor pre-synchronization activity. Therefore, we aim to reduce the incidence of this type of misprediction. Note that technically, this kind of misprediction actually saves *more* energy because it runs the processor in Idle for longer (see the blue arrow in Figure 6). However, since it misses important pre-synchronization activity, this type of energy saving is actually undesirable. Overall, we use low power modes to save energy, running the risk that we occasionally mispredict neuronal activity and lose some pre-synchronization activity. But if we reduce the incidence of this type of misprediction, this tradeoff is worthwhile since we ultimately sustain far longer battery life and collect considerably longer neuronal activity recordings overall.

④ Incorrectly predicted synchronization: Finally, the branch predictor may incorrectly predict synchronized behavior, only to find that this behavior does not occur in the next epoch. This represents wasted energy usage as the processor is transitioned to Nominal operation unnecessarily. However, as soon as we detect no Purkinje synchronization in the following epoch, we transition the processor back to Idle mode.

In Figure 5, the branch predictor predicted, in B, that the upper left neuron would fire in C. Ultimately the lower right neuron fired. We refer to such predictions as *accidentally correct* as they represent situations where prediction of synchronization is correct even though the prediction of the individual Purkinje neurons are wrong. While accidentally correct predictions enable correct operation, our goal is to design predictors that are correct in a robust manner, and do not rely on "accidental luck". We therefore focus on accuracy for both per-neuron and synchronized prediction.

4.2 Branch/Brain Predictor Implementation

Our modifications leave branch predictor access latencies, energy, etc., *unchanged* in normal operating mode. Therefore, this section focuses on neuronal prediction in low power mode. Figure 7 presents our proposed hardware. On the left, we show a mouse with an embedded implant. Purkinje activity is digitized by the ADC, and stored in a designated DRAM location called an activity buffer.

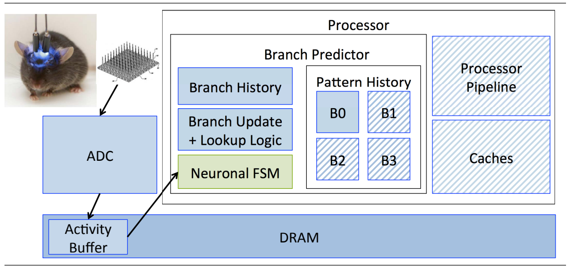

Figure 7: In idle low power mode, striped components are powered off, while a hardware FSM co-opts (part of) the branch predictor for neuronal prediction. Components are not drawn to scale.

An important first step in managing the Utah array is to identify which conductive silicon needles on the array correspond to Purkinje neurons. Recall that the Utah array has hundreds of needles. Many of them probe non-neuronal tissue, while others probe neurons. Implants, therefore, run calibration code on installation to associate needles to specific neurons by studying 1-2 seconds of neuronal activity [33]. Since the implant stays in place, once calibration completes, we know exactly which of the Utah array needles correspond to Purkinje neurons.

Figure 7 shows that after calibration, when the processor is placed in low power mode, the pipeline and caches are gated off (indicated by stripes). However, the branch predictor is treated differently. We show a branch predictor structure made up of pattern history tables and branch history tables[3]. These branch predictor structures are looked up and updated using combinational logic.

When the processor is in idle low power mode, the branch predictor is used to perform neuronal prediction. One option is to leave the entire branch predictor structure on for this purpose. However, this is needlessly wasteful since modern branch predictor tables tend to use tens of KBs with thousands of entries. Meanwhile, modern recording technologies allow us to probe the activity of hundreds of neurons simultaneously [63] so we only technically require hundreds of entries in the branch predictor to make per-Purkinje spike predictions. Therefore, we exploit the fact that modern branch predictors are usually banked [8, 55] and turn off all but one bank. This bank suffices to perform neuronal prediction. To enable this bank to remain on while the remainder of the branch predictor is power gated, we create a separate power domain for it. This requires a separate set of high Vt transistors and control paths. We model the area, timing, and energy impact of these changes (see Section 7).

Figure 7 shows that we add a small neuronal FSM (in green). We modify the code run in the calibration step after implant installation to add a single store instruction. This updates the contents of a register in the neuronal FSM maintaining a bit-vector used to identify which of the Utah array's silicon needles probe Purkinje neurons. The neuronal FSM uses this bit vector to decide which entries in the activity buffer store activity from neuronal (rather than non-neuronal) tissue. The neuronal FSM then co-opts the branch

[3]While our example shows one branch history register, the same approach could be applied to branch history tables too.

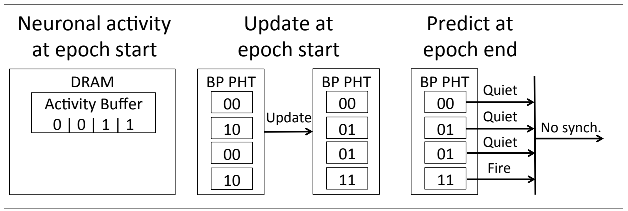

Figure 8: In low power mode, the branch assesses the likelihood of neuronal spiking in the next epoch. We show the Smith predictor as an example.

predictor for neuronal prediction during idle low power mode. It guides two types of operations, every time epoch:

① Updates with neuronal outcomes: In every epoch, we first update predictor tables with recent Purkinje activity. Consider Figure 8, which shows the DRAM activity buffer at the start of the epoch. The activity buffer maintains an entry for every conductive needle in the Utah array indicating firing (a bit value of 1) and quiescence (a bit value of 0). Our example shows entries for four conductive needles probing four neurons, two of which remained quiet and two of which fired at the start of the epoch. Consequently, the neuronal FSM updates the branch predictor bank left on by treating each neuron as a separate branch and updating in a manner that mirrors conventional branch predictor updates. Figure 8 shows this for a Smith branch predictor with 2-bit saturating counters and hysteresis [60]. The four branch predictor entries are used for neurons 0-3 and are updated using the state machine of a standard 2-bit branch predictor.

② Predictor lookups for neuronal predictions: Figure 8 shows that at the end of the epoch, the neuronal FSM predicts whether Purkinje synchronization will occur in the next epoch. Each neuron's branch predictor entry is looked up to predict whether that neuron will fire. In our example, the first three neurons are predicted to remain quiet while the last one is predicted to fire. Combinational logic assesses whether enough neurons are predicted to fire to constitute synchronization. For our example in Section 4.1, where at least two neurons must fire for synchronization, the neuronal FSM assesses that the next epoch will not see synchronization and hence the processor can continue in idle low power mode.

While we do not change access times for branch prediction, we consider timing when the branch predictor is used for neuronal prediction. Using detailed circuit modeling, we have found that since neuronal spiking times (and epoch times) range in the order of milliseconds, the timing constraints of modern branch predictors, which are typically designed for hundreds of MHz and GHz clocks, are easily met.

Finally, one may consider sizing the DRAM activity buffer to be large enough to store lead-up activity to synchronization. The processor could be transitioned to nominal operation when synchronization occurs. This approach seemingly preserves lead-up activity to synchronization without needing synchronization prediction. However, while this approach suffices for some implants, it is not a solution for implants that cannot defer neuronal processing, like the ones that provide stimuli to the brain immediately when a specific neuron fires [52, 72]. Our goal to enable energy management on *all* implant types – therefore, we tackle the harder problem of neuronal prediction.

4.3 Saving Predictor State

In our design, the branch predictor oscillates between conventional branch prediction and neuronal prediction. The advantage of this approach is that it is relatively simple to implement. The disadvantage, however, is that we lose predictor state when we switch between power states. This can be harmful to both branch prediction and neuronal prediction. For branch prediction, the impact is reminiscent of context switching [26]. We have found that compared to a baseline that does not use idle low power modes, we lose 3% average branch prediction accuracy. Fortunately, this does not have a discernible performance impact on the workloads run on our implant. However, the other overhead of losing neuronal prediction data does have an effect. We find that neuronal prediction accuracy, particularly when the mouse is awake and exhibits more sophisticated neuronal firing patterns, drops by roughly 7%. We have therefore also studied the benefits of augmenting the baseline approach with mechanisms whereby predictor state is saved when transitioning the core from low power mode to nominal operating frequency. Specifically, when this transition occurs, the Neuronal FSM reads all the branch predictor contents and writes it to a reserved portion of DRAM. Later, when the core is transitioned to low power mode, the Neuronal FSM reads this state back into the predictor. We have modeled this approach and found that it almost doubles transition time from/to low power state; however, since this increased time continues to remain in the microsecond-time range, it is far lower than 10ms epochs and does not affect operation adversely.

Despite its benefits, this approach cannot learn neuronal patterns while the core runs in nominal operating mode. If there are phase changes in neuronal firing patterns between when the core was raised to nominal operation and when it returns to low power mode, these are not reflected in the saved neuronal predictor state that is recovered on entry into low power mode. Despite this drawback, Section 8 shows that just saving state improves neuronal prediction accuracy and saves 5% average additional energy.

5 BRANCH AND BRAIN PREDICTORS

Though broadly studying all branch predictors is beyond the scope of this paper and needs further work, we focus on:

Smith predictors: These use 2-bit saturating counters with hysteresis (see Figure 8). Each Purkinje neuron is allotted a counter in the prediction table[4]. A branch/neuron's local history is used to predict future behavior (i.e., correlations among branches/neurons are not exploited). We have found that local history can, to some extent, enable prediction of future activity. But, this approach is too simple to accurately predict spiking when the mouse roams around and hence sees more complex cerebellar spiking.

Gshare predictors: Purkinje neurons often form "micro-bands" or groups where neurons fire close together in time in a synchronized manner [53, 65]. To exploit micro-bands, we study branch

[4]Since branch predictors use large tables with far more entries than the tens-hundreds of neurons we can currently record, we assume one entry per neuron and no aliasing.

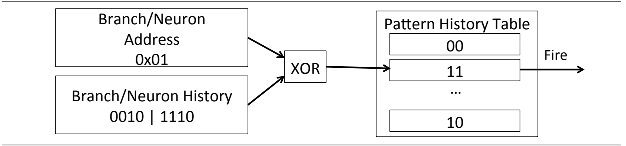

Figure 9: Adapting gshare predictors for neuronal prediction.

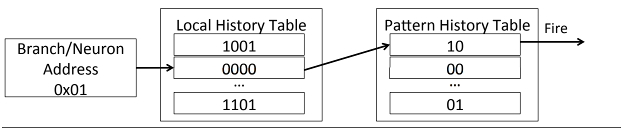

Figure 10: Adapting two-level predictors for neuronal prediction.

predictors that exploit correlated branches. Gshare is a well-known example of such a predictor. Figure 9 shows how gshare predictors can be co-opted for neuronal prediction. The neuronal FSM from Section 4 looks up the predictor table for each individual neuron. If enough of them are predicted to fire, a synchronization event is predicted. Figure 9 illustrates lookup for neuron number 1. Gshare performs an exclusive-or between this “address” (or neuron number) and an n-bit global history register, which records the last n branch outcomes globally. For neuronal prediction, one could similarly record spiking behavior of the last n neurons. There are two options – (a) we can record whether there was Purkinje synchronization in the last n epochs (one bit per epoch); or (b) whether each of j individual neurons in the last k epochs (where n equals j×k) fired or not. Recall that our goal is to perform accurate per-neuron predictions, not just synchronization predictions (see Section 4.2). We therefore do take the second option, (b). Figure 9 shows a global history register that stores activity from four neurons in the last two epochs.

Two-level adaptive predictors: Two-level adaptive predictors exploit inter-branch correlations with global history, but also with per-branch histories [71]. Figure 10 co-opts this approach for neuronal prediction, focusing on the lookup for neuron 1. The neuron number, like the branch address, indexes a table of local history registers. The n-bit registers record outcomes of the last n branches and neuronal data that map to that location. In Figure 10, the local history tables store information on how neuron 1 spiked in the last four epochs (in our example, neuron 1 was quiet in all four epochs). This history selects a pattern history table entry, which is used to predict neuron 1's activity in the next epoch.

Perceptron predictors: Perceptrons are best able to leverage inter-branch/neuron correlations. Figure 11 illustrates the operation of a perceptron predictor and shows how we can adapt it for neuronal prediction [26]. A table of perceptrons is looked up for each branch or neuron. Each perceptron entry maintains weights for each correlated branch/neuron. Like branch prediction [24, 26], we use the weights and prior branch outcomes, stored in the history register, to calculate:

$$y = w_0 + \sum_{i=1}^{n} x_i w_i$$

Here, w_0 is a bias weight, w_i are weights for correlated branches/neurons, x_i are prior branch/neuron outcomes, and y is the output prediction. If y is non-zero, the branch/neuron is predicted

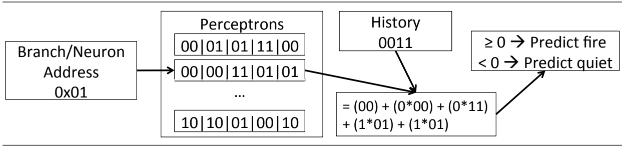

Figure 11: Adapting perceptron predictors for neuronal prediction.

taken/fired. In Figure 11, we use 2-bit weights though actual implementations use 8-bit integer weights [24–26]. The weights record a branch/neuron's dependence on its past behavior through a bias weight, and its dependence on other (four other, in our example) branches/neurons through other weights. All values are stored in one's complement, like the original design [26], with large positive and negative values indicating positive and negative correlations respectively. Figure 11 shows that the looked-up neuron is weakly correlated with its past (a bias weight of 00) but is positively correlated with neurons 2 and 3 (weights of 01), and strongly but negatively correlated with neuron 1 (weight of 11).

During neuronal prediction mode, the neuronal FSM first trains the predictor with the spiking outcomes of the current epoch. In order to perform the update, each neuron's perceptron entry and the global history register (which maintains the spiking outcomes of the last epoch) are used to re-calculate the perceptron's prediction for this epoch. When the neuron's prediction from the last epoch does not match the outcome in the current epoch or if the weighted sum's magnitude is less than a threshold θ (used to gauge if training is complete), the perceptron entry is updated as per usual. The algorithm increments the ith weight if the branch/neuron outcome agrees with x_i and decrements it otherwise. We assume the θ values used in prior work for branches [26] as they suffice for neuronal prediction too. Once this is done, the neuronal FSM replaces the global history register contents with the outcomes of the most recent activity in the activity buffer. Subsequently, each perceptron entry is read and using the new outcomes, a weighted sum is calculated, and a neuronal prediction is made.

Neuronal prediction treats the global history buffer differently from conventional branch prediction, where the first outcome is associated with the most recent branch in time, the second outcome with the second most recent branch in time, and so on. With neuronal prediction, there is no time-ordering among neurons in an epoch. This does not present correctness problems because the neuronal FSM first uses the global history buffer values (holding spiking from the previous epoch) to update all perceptron entries at the start of the current epoch. It then updates the global history buffer with current spiking outcomes and uses it with every perceptron entry to make a prediction for every neuron.

As perceptron size scales linearly with the number of correlated branches/neurons, they exploit longer branch/neuron correlation histories than other schemes, which scale exponentially. This makes perceptrons effective at capturing Purkinje micro-bands. Moreover, the two traditional problems with perceptrons – access latency and power consumption [5, 25] – are less of a concern in our design. Access latencies are usually a problem on high-performance GHz-range processors with tight cycle times. Instead, our implanted processor requires branch predictions on a 300MHz clock or neuronal prediction in 10ms epochs, which perceptron predictors using Wallace tree adders [27] can comfortably achieve. And despite

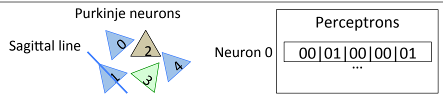

Figure 12: Parasagittal neurons spaced a few micron apart are usually correlated. This is reflected in their perceptron table entries.

the higher energy requirements of perceptron predictors, their ability to accurately predict neuronal activity enables the more aggressive use of low power modes and hence much lower overall system energy. Finally, although Figure 11 shows perceptrons with global history, we also study two-level approaches, where a table of per-branch/neuron histories finds the desired perceptron.

5.1 Lessons Learned

In using branch predictors to perform neuronal prediction, we have learned the following:

① **Correlations matter more than local history:** A neuron's history can provide some indication of future behavior. But local history must be coupled with inter-neuronal correlations for good prediction accuracy. This is because micro-bands of correlated neurons synchronize [53], and predictors that can exploit longer histories of correlated neurons are hence most accurate. Area-equivalent perceptron predictors can achieve 35% prediction accuracy over Smith predictors.

② **Correlations remain important in the presence of sensorimotor stimulation:** When we blow air puffs on the whiskers of anesthetized mice or study free-roaming mice, Purkinje activity is often correlated and synchronization becomes more frequent. Smith predictors, which rely on only local history, drop off in accuracy. For example, awake mice see an average of 27% accuracy, while gshare and two-level adaptive approaches achieve average accuracies of only 35%. Perceptrons continue to exploit inter-neuronal correlations and achieve much better accuracy. We also qualitatively observe that when awake mice move more, perceptron predictors are more accurate than other approaches.

③ **Prediction accuracy trumps higher energy needs:** Two-level adaptive and perceptron approaches consume more power than simpler Smith predictors. We find, however, that complex predictors, especially perceptrons, predict neuronal activity so much more accurately that they can use low power modes aggressively enough to save energy overall.

④ **Neurons experience "phase changes":** Branch mispredictions often occur when branch correlations change, or when branches are not *linearly separable* for perceptrons. Linear separability refers to the fact that perceptrons can perfectly predict only branches whose Boolean function over variables x_i have its true instances separated from its false instances with a hyperplane, for some values of w_i [27]. Similarly, there are situations when perceptron predictors achieve only 30% neuronal prediction accuracy (and other predictors achieve even less) because many *neurons* are not linearly separable. This is because neurons, just like branches, exhibit phase-like behavior. Groups of Purkinje neurons sometimes switch between micro-bands – i.e., a neuron changes which other neurons it correlates with. This well-known biological phenomenon [53] can lead to mispredictions. We will explore techniques like piecewise linear branch prediction, which target linear separability of branches [25], to overcome this problem for neurons in the future.

⑤ **Predictors can capture brain physiology:** Parasagittal neurons spaced micrometers apart are known to experience correlated spiking [53]. Figure 12 shows that neurons have a sagittal line dividing their bodies into equal left and right sides. Parasagittal neurons are those that are parallel to one another's sagittal lines. In our example, neurons 0, 1, and 4 are parasagittal and correlated. We have found that perceptron branch predictors accurately capture correlations among parasagittal Purkinje neurons, maintaining much larger weights for them. On average, the weights for parasagittal neurons are 50%+ larger than the weights for other neurons. Figure 12 shows an example where the weights for neurons 1 and 4 are positively correlated in neuron 0's perceptron.

6 OTHER IMPLEMENTATION STRATEGIES

Our goal is to show how neuronal prediction can be performed using existing hardware to save energy. To do this, we have built a proof-of-concept system. However, other design options exist:

Other predictors: While we co-opt well-known branch predictors, the higher-level observation is that it is generally possible to harness learning hardware that is sufficiently efficient for implementation on modern chips, for neuronal prediction. It may be that richer machine learning hardware for concepts like cache line reuse [68] or more sophisticated hardware neural networks [13] may be effective too.

Using a separate dedicated hardware block: We modify the existing core and co-opt its branch predictor for neuronal prediction, but this is not the only design option. In fact, it may be fruitful to instead embed an additional IP core for power management of the embedded processor. While this does mean that we would need more hardware and silicon for the IP block, it would also leave the embedded core largely untouched. Furthermore, as our understanding of the brain deepens, a separate IP block may make it easier to upgrade learning hardware for neuronal prediction, without changing the embedded processor. Finally, a problem with cannibalizing the embedded processor's branch predictor is that we cannot train the predictor with neuronal activity when the processor remains in nominal operating mode (see Section 4.3). We have run experiments to quantify the associated loss in neuronal prediction accuracy and have found average losses of 3% accuracy for anesthetized mice and 5% for awake mice. A separate IP block can sidestep this problem. Our study paves the way for future work on this alternative approach.

7 METHODOLOGY

Simulation infrastructure: This paper performs early-stage design exploration. Therefore, rather than implement the chip in hardware, we rely on careful cycle-accurate software simulation. We model a processor similar to the ARM Cortex M7, with the configuration of Table 1. Our processor runs at 300MHz and uses the standard Cortex M7 idle low power mode where pipelines and caches can be gated off to save power. We use CACTI [48] and McPAT [36] for power/energy analysis. We model the area, timing, and energy

Pipeline	2-issue, 6-stage, in-order, forwarding
Instruction and data cache	32KB with ECC
Baseline branch predictor	8KB Smith predictor
Integer/FPU	4-stage/5-stage pipe
Register file	6/4 read/write ports

Table 1: Parameters of our system.

implications of creating a separate power domain for the branch predictor bank for neuronal prediction. The additional Vt transistors and control wiring/logic increases chip area by 1.4%. Branch prediction access latencies remain unchanged, however. Further, we model the neuronal FSM. In general, we find that its simplicity means that it can be implemented with area-efficient and energy-efficient combinational logic.

Workloads: We use four neuronal spiking analysis workloads, selected for their common use in the neuroscience community, to extract biologically relevant data from neuronal recordings [33]. The four workloads are:

① Compression: We use **bzip2** to compress the spiking data recorded by the ADC for 500ms after synchronization.

② Artifact removal: Microelectrode arrays can pick up noise from muscle movement in the scalp, jaws, neck, body, etc. These artifacts can be removed with principal component analysis. Our **pca** benchmark stacks the data from our electrodes, and for each electrode, projects into the PCA domain, yielding cleaned signals [33].

③ LFP extraction: In **lfp**, we apply a fifth-order Butterworth filter on the neuronal data to enhance low-frequency signals in the range of 0.5-300Hz, as is common practice [33].

④ Denoising: Reducing noise in neuronal recordings is an important step in neuronal processing. There are several ways to denoise, but we use discrete wavelet transforms or **dwt** with Rigrsure thresholding, similar to prior work [33].

Mouse surgeries: We extract neuronal activity from mice in vivo using state-of-the-art optogenetics. Optogenetics gives neuroscientists the ability to use pulses of light to image and control almost any type of neuron in any area of the brain, with precise timing. We perform surgeries on C57BL/6 mice on postnatal days 21-42. We perform small craniotomies of approximately 2mm diameter over lobule 6 locations on the mice cerebella, from which we collect Purkinje activity. For mice under anesthesia, we use ketamine/xylazine to achieve deep anesthetized state. Further, we load the Purkinje cells of the area of study with calcium indicator Oregon Green BAPTA-1/AM (Invitrogen), as described previously [65]. This indicator fluoresces under exposure to light, allowing us to collect images such as Figure 3 using two-photon microscopes [53]. We track the activity of 32 Purkinje neurons.

8 RESULTS

Per-neuron prediction accuracy: We quantify per-neuron prediction accuracy. We study the cross-product of recordings on awake (Awake) and anesthetized mice (with air puffs (Anesthesia-stimulus) and without (Anesthesia-No-Stimulus)) and our four benchmarks, totaling 52 experiments. Note that these results assume that neuronal is *not* saved when the processor is transitioned from low

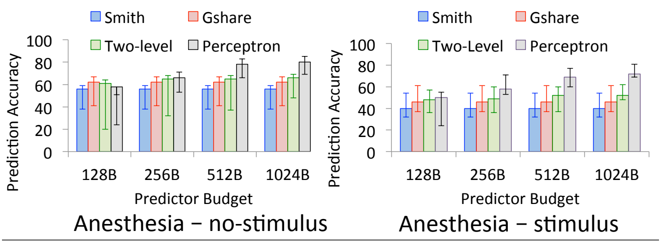

Figure 13: Prediction accuracies for mice under anesthesia without stimulus, and with air puffs blown into their whisker pads. We vary the hardware budget available for branch predictor bank left open.

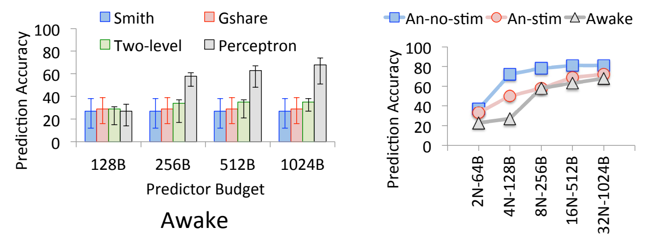

Figure 14: (Left) Prediction accuracy for awake free-roaming mice, as a function of the predictor area budget; and (right) perceptron predictor accuracy as a function of the neuron history length.

power to nominal mode. We show results when we save state at the end of this section.

Figure 13 presents results for anesthetized mice. The y-axis plots per-neuron prediction accuracy. The x-axis shows the hardware budget for one predictor bank, which is all we need for neuronal prediction. Modern branch predictors are 8-16KB, and 1KB banks are reasonable. For each hardware budget, we have exhaustively studied predictor organizations and report results from the organization with the best average accuracy. At each hardware budget, we find that gshare and two-level predictors perform best when they maintain history for 0.5-0.6× the neurons as the perceptron.

Figure 13 shows that perceptrons predict neuronal activity more accurately than other approaches, particularly with larger budgets. Smith predictor accuracies flatten as the smallest size (128 bytes) has enough area to maintain per-neuron counters. But perceptrons require more space, so they benefit from 1KB budgets. Larger budgets also permit better gshare and two-level adaptive predictor accuracy. At modest hardware budgets of 1KB, perceptron predictors achieve an average prediction accuracy of 80%, and as high as 85%. Perceptrons become even better than other approaches when sensorimotor stimulation, and hence the complexity of Purkinje activity, increases (see Anesthesia-stimulus results).

The left side of Figure 14 shows results for awake mice. The increased complexity of spiking prompts Smith, gshare, and two-level adaptive predictors to mispredict more often but perceptrons still achieve an average of 68%. We found that prediction accuracy varies more when the mouse moves its tail and also its limbs (see the larger min/max bars).

The graph on the right of Figure 14 shows how perceptrons achieve more accuracy. We show accuracy as we vary the number of weights stored in each perceptron entry. A label of jN-kB on the x-axis indicates 8-bit integer weights for j potentially correlated neurons, totaling k bytes (assuming that we need a separate

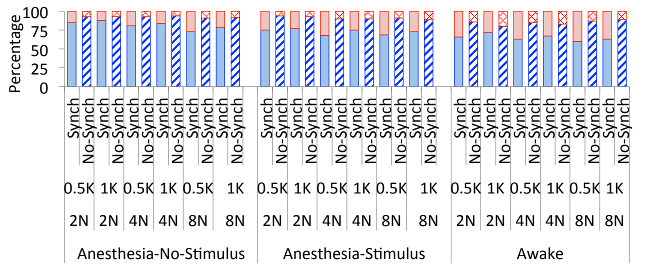

Figure 15: Percentage of synchronized events predicted correctly (solid blue) and incorrectly (solid red), and percentage of unsynchronized events predicted correctly (striped blue) and incorrectly (striped red). We show average results and vary the number of neurons in a synchronized event from 2, 4, to 8, and the predictor size between 512 bytes and 1KB. All results are for perceptron predictors.

perceptron entry for every neuron we want to predict). The larger k is, the greater the correlation history amongst branches/neurons, and the more accurate our neuronal prediction. When we plotted this data, we noticed an interesting relationship between the number of weights required for accurate predictions and the biology of Purkinje neurons. Studies have shown that usually, a handful (2 to 8) of neurons form micro-bands [53]. The graph on the right of Figure 14 mirrors this observation, with sharp accuracy benefits when the number of weights in the perceptron goes from 2 to 8, particularly when the mouse is awake.

Synchronization prediction accuracy: So far, we have discussed prediction accuracy for each individual Purkinje neuron's behavior. However, our aim is to ultimately predict synchronization behavior. We focus on the perceptron predictor for these studies as they are far more accurate than other approaches. While good prediction accuracy for individual neurons is a good indicator of synchronization prediction, their relationship is complicated by two competing factors. On the one hand, accidentally correct predictions may occur (see Section 4.1), boosting synchronization prediction accuracy. On the other hand, synchronization requires *multiple* neurons to be simultaneously predicted correctly. The probability that multiple neurons are concurrently predicted accurately is lower than accuracy for an individual neuron.

Figure 15 summarizes synchronization prediction accuracy. We separate results for awake and anesthetized mice, varying the perceptron predictor hardware budget between 512 bytes and 1KB. We vary the number of neurons that must simultaneously fire to be considered synchronized from 2 to 8 (represented by 2N, 4N, and 8N). We plot two bars, separating correct/incorrect predictions for synchronized and non-synchronized events.

Figure 15 shows that perceptrons accurately predict most synchronized and non-synchronized events. Accuracy increases with larger predictors, but remains consistently 75%+ under anesthesia with no stimulus. Naturally, stimuli and awake states make prediction harder, but perceptrons still consistently predict correctly more than 60% of the time. We also find that accuracies diminish as synchronization thresholds vary from 2 to 8 neurons. The higher the threshold, the more the number of neurons that have to predicted correctly. Regardless, prediction accuracy remains high.

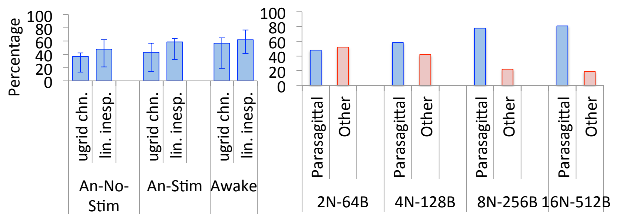

Figure 16: (Left) Percentage of Purkinje neurons that experience micro-grid changes (ugrid-chn.**) and are linearly inseperable (**lin.-insep.**) with averages, min/max shown; and (right) for each neuron predicted to fire, percentage of total weighted sum value contributed by the weights from parasagittal neurons versus others.**

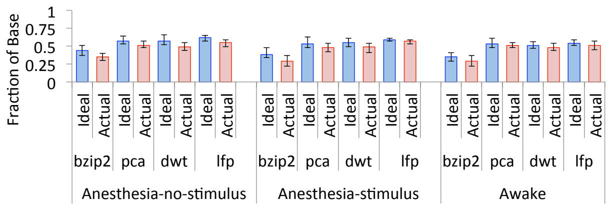

Figure 17: Fraction of baseline energy saved using Ideal **and** Actual **prediction. We assume perceptrons with 32 8-bit weights (1KB budget) and that 4 neurons must fire to be considered synchronized.**

Understanding prediction characteristics: We now discuss the source of mispredictions, focusing on perceptrons as they predict neuronal behavior most accurately. Like branch misprediction, most neuronal misprediction arises from neurons that are linearly inseparable. Past work identifies the fraction of static branches that are linearly inseparable to understand mispredictions [26]. The graph on the left in Figure 16 does the same, but for neuronal prediction (lin.-insep.). There is a biological basis for linear inseparability – neurons sometimes change which other neurons they correlate with. We study our neuronal traces and every 10ms, identify micro-grids. As a fraction these samples, we plot the percentage of time that neurons change between micro-grids (ugrid chn). Figure 16 shows that adding sensorimotor stimulation (An-Stim and Awake) increases micro-grid switches and linearly inseparable neurons, lowering prediction accuracy.

The graph on the right in Figure 16 shows that perceptrons also accurately capture the biology of parasagittal correlations. Every time the predictor predicts spiking, we log what percentage of the perceptron's weighted sum originates from weights of parasagittal neurons. The higher the percentage, the higher the correlations between parasagittal neurons. We plot these percentages in blue (with the rest shown in red), as a function of the perceptron predictor size and the number of weights in each perceptron entry (e.g., jN-kB indicates j weights and k bytes). As expected, with more weights, parasagittal correlations are more easily tracked.

Global versus global/local perceptron histories: Beyond perceptrons with global history, we have also studied a mix of local and global history [27]. Because prediction accuracy hinges on neuronal correlations, we see little benefit (less than 1% more accuracy) from the addition of local history.

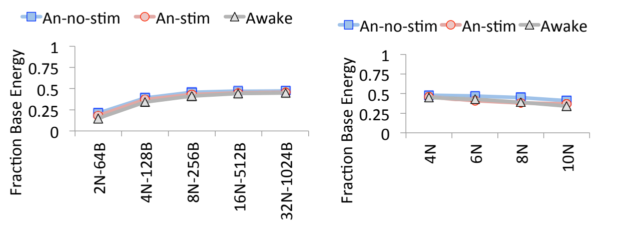

Figure 18: (Left) Average fraction of baseline energy saved when using a perceptron predictor, for different numbers of weights. We assume that 4 neurons must fire to be considered synchronized; and (right) average energy saved when using a perceptron predictor with 32 8-bit weights (1KB total budget) and varying the number of neurons that must fire to be considered synchronized from 2 to 10.

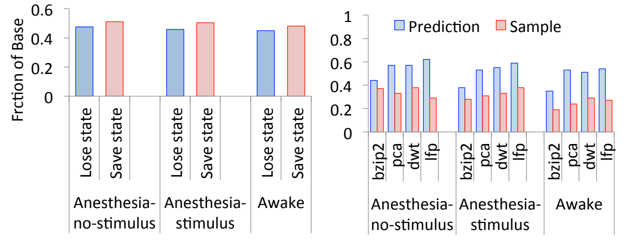

Figure 19: (Left) Increased energy savings using perceptron predictors with 1KB budget and assuming synchronization threshold of 4 neurons, when saving predictor state (red bars) versus not saving state (blue bars); and (right) fraction of baseline energy saved using neuronal prediction (blue bars) versus sampling (red bars), assuming that both approaches record an equal number of pre-synchronization events.

Energy savings of perceptrons: Figure 17 quantifies the fraction of energy saved versus the baseline described in Section 3. We discuss the energy benefits of using Smith, gshare, and two-level adaptive predictors, but for now focus on perceptrons as their energy benefits far outweigh the other approaches. Figure 17 assumes 1KB perceptron predictors and that four neurons must fire close together in time to be considered a synchronized event.

Figure 17 shows that our energy savings (Actual) are within 5-10% of the Ideal savings from oracular prediction. Overall, this corresponds to energy savings of 22-59%. Even with stimuli, which decreases energy saving potential since there are more synchronized events, our approach saves 22-50% energy on Awake mice.

Figure 18 sheds more light on energy trends. The graph on the left shows the energy saved as the number of 8-bit weights per perceptron entry varies from 2 to 32. More weights improve predictor accuracy by capturing micro-grid correlations. Increasing the number of weights from 2 to 8 doubles energy savings for anesthetized and awake mice. The graph on the right shows the average energy saved by a 1KB perceptron predictor (with 32 weights per entry), as we vary the number of neurons that must concurrently fire to be considered a synchronized event. As this number increases, energy savings decrease as there are fewer instances of synchronized events. Nevertheless, even when we assume that 10 neurons must fire to constitute synchronization, we save an average of 30%+ of energy. And since scientists generally study micro-grids of 4-8 neurons [53], average energy savings are closer to 45%+.

Undesirable energy savings: In Section 4.1, we discussed situations where the predictor incorrectly predicts no synchronization. This mistake prompts loss of pre-synchronized activity, so its energy savings are undesirable. We have found that less than 2% of total energy savings for our workloads are undesirable in this way. The reason this number is small is that perceptrons have good prediction accuracy. The (few) mispredictions idle the processor for an extra 10ms (the time taken to identify the misprediction). Subsequently, the processor transitions to nominal operation. Compared to the long stretches of times that the processor is correctly predicted and placed in idle low power mode (10s of seconds), these mispredictions minimally affect energy saved.

Energy savings versus other branch predictors: In Section 3, we showed that when low power modes are not used, the embedded processor consumes 5% more core energy when using perceptrons versus Smith prediction. However, when the perceptron guides power management, we find that superior accuracy consistently yields 5-45% energy savings compared to Smith, gshare, and two-level adaptive approaches.

Impact of saving predictor state: Section 4.3 showed that it may be useful to save predictor state when transitioning from low power to nominal mode, so that whatever the predictor has learned about neuronal firing patterns is not lost and the predictor does not have to be retrained from scratch. The graph on the left of Figure 19 quantifies the increase in energy savings that saving predictor state provides, assuming perceptrons and 1KB sizes for neuronal prediction. We show average energy savings when state is lost (the blue bars, which correspond to the results we have discussed in this section so far) and when it is saved (the red bars). In all cases, the increased accuracy of the perceptron predictor allows it to identify more opportunities to suspend the processor in idle low power mode, saving an additional 5% energy on average across the board.

Sampling techniques: Neuronal prediction: ① captures most of the pre-synchronization activity; ② captures all synchronization activity, even when it misses pre-synchronization, because the neuronal FSM transitions the processor to nominal frequency on mispredictions; and ③ achieves good energy savings. Deterministic sampling techniques cannot guarantee capture of all synchronization events (unless it also keeps track of neuronal activity with something akin to a neuronal FSM), or all pre-synchronization events. Nonetheless, we assess how deterministic sampling fares. We model deterministic sampling techniques, where we wake up the processor every Nth epoch. We vary N exhaustively from 1 to as many epochs as in our trace. On a sample, if synchronization is detected, we begin processing 500ms of the spiking data. If no synchronization is detected, however, we cannot immediately return the processor to low power state as there is no way to tell if the neurons are leading up to synchronization. Therefore, we still kept the processor on for the epoch in case synchronization begins at its end. We also model exponential backoff sampling, where the absence of synchronization in a sample means that the wait time before the next sample is twice the last number of waited epochs.

Through exhaustive search, we found that it is possible for sampling to record (close to) as many lead-up and synchronization samples as prediction, but with a much higher energy overhead. The key culprit is the need to conservatively leave the processor on for the entire sample epoch. The graph on the right of Figure 19 compares the energy savings of neuronal prediction versus the sampling approach with matching pre-synchronization coverage. Generally, sampling loses 10-20% energy savings compared to neuronal prediction. Nevertheless, we believe that it may be possible to combine both approaches going forward. The challenge will be to ensure that sampling remains sufficiently simple that it is worth doing rather than augmenting the neuronal predictor.

Energy savings with dynamic voltage/frequency scaling: We have focused on idle rather than active low power modes like dynamic voltage/frequency scaling (DVFS) because Cortex M processors currently support only idle modes. However, we were curious about energy savings if DVFS were to be incorporated. We studied and compared three schemes using a 1KB perceptron predictor: (1) use idle low power modes as described thus far; (2) use DVFS instead of neuronal prediction, to identify opportunities when the workloads can afford to be slowed down to 0.5× and 0.75× of the baseline frequency using standard CPU utilization based prior approaches [15]; and (3) combine (1) and (2) by showing that DVFS and idle mode usage with neuronal prediction are orthogonal. We found that (2) remains superior to (1), saving an average of 12% more energy. Combining neuronal prediction and idle low power modes with DVFS promises even more energy savings (as high as 67%, exceeding the 59% of neuronal prediction alone).

9 RELATED WORK

This work adds to the growing body of recent literature on implantable brain-machine interfaces. While recent studies propose implants with embedded processors and study energy/power management, we present the first study on topics pertaining to the processor architecture in these devices [6, 20, 38, 47, 64]. In the context of studies from the architecture domain, our work relates to hardware neural network accelerators [4, 12, 13, 37], but is closer to studies linking neural biology with architecture. Hashmi et. al. studied fault tolerance in cortical microarchitectures [21] while Nere et. al. emulated biological neurons digitally [49]. Their work paved the way for Smith's pioneering studies on efficient digital neurons for large-scale cortical architectures [61]. We are inspired by these studies but co-opt *existing* hardware structures to predict neuronal activity and manage energy.

10 CONCLUDING REMARKS

Generality of observations: How conclusively can we say that branch predictors, or indeed any hardware predictors, can predict brain activity? To answer this question, we need to study much more than just 26 minutes of neuronal activity, from more locations than just lobule 6 of the cerebellum. This work is just a first step in this research endeavor.

Broader implications: Neuronal prediction is generally useful for implants. For example, designers of deep-brain stimulation implants want to predict pathological neuronal activity, so that they can stimulate portions of the brain like the thalamus pre-emptively to mitigate epilepsy symptoms [59]. This approach saves implant energy over current approaches which do not know when neuronal activity of interest is likely to occur and hence continually monitor the brain instead. Similar approaches are applicable to hippocampal implants for memory enhancement [57]. Consider also implants which dynamically switch between compression algorithms to record neuronal activity [58]. Neuronal prediction can enable quicker and better choices about which compression algorithm to use before the advent of the activity.

Relationship with neural circuits: Neuroscientists are seeking to model neural circuits that explain neuronal biology [28]. It may be fruitful to consider whether models of well-known architectural structures like branch predictors could aid neural circuit modeling, particularly if they predict neuronal activity accurately.

11 ACKNOWLEDGMENTS

We thank Jonathan Lovelace for help with the wet-lab portion of this work, as well as Guilherme Cox, Ján Veselý, Zi Yan, Daniel Lustig, Carole-Jean Wu, and Tushar Krishna for feedback on early drafts of this manuscript. We thank James Smith for discussions on future extensions of this work. This work was made possible in part due to NSF's award number 1319755. Finally, we are grateful to Shampa Sanyal for inspiring us and making this work possible.

REFERENCES

[1] [n. d.]. Cerebellum Image. *Life Science Databases/Wikimedia Commons* ([n. d.]).
[2] [n. d.]. Utah array. *http://churchlandlab. neuroscience.columbia.edu/images/utahArray.jpg* ([n. d.]).
[3] [n. d.]. Utah array plugged. *http://prometheus.med.utah.edu/b̃wjones/wp-content/uploads/iblog/Implant1.jpg* ([n. d.]).
[4] Manor Alwani, Han Chen, Michael Ferdman, and Peter Milner. 2016. Fused-Layer CNN Accelerators. *Int'l Symp. on Microarch.* (2016).
[5] Renee St. Amant, Daniel Jimenez, and Doug Burger. 2008. Low-Power, High-Perf. Analog Neural Branch Prediction. *Int'l Symp. on Microarch.* (2008).
[6] Gian Nicola Angotzi, Fabio Boi, Stefano Zordan, Andrea Bonfanti, and Alessandro Vato. 2014. A Programmable Closed-Loop Recording and Stimulating Wireless System for Behaving Small Laboratory Animals. *Scie. Reps.* 4, 5963 (2014).
[7] M Aravind and Suresh Babu. 2015. Embedded Implementation of Brain Comp. Interface Concept Using FPGA. *Int'l Jnl. of Science and Research* (2015).
[8] Amirali Baniasadi and Andreas Moshovos. 2002. Branch Predictor Prediction: A Power-Aware Branch Predictor for High-Perf. Processors. *ICCD* (2002).
[9] Boris Barbour. [n. d.]. Equipe Cervelet. *http://www.ibens.ens.fr* ([n. d.]).
[10] Abhishek Bhattacharjee and Margaret Martonosi. 2009. Thread Criticality Predictors for Dynamic Perf., Power, and Resource Management in Chip Multiprocessors. *Int'l Symp. on Comp. Arch.* (2009).
[11] Michael Bielawski and Helen Bondurant. 2015. Psychosis Following a Stroke to the Cerebellum and Midbrain: A Case Report. *Cerebellum and Ataxias* (2015).
[12] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2014. DaDianNao: A Machine-Learning Supercomputer. *Int'l Symp. on Microarch.* (2014).
[13] Yu-Hsin Chen, Tushar Krishna, Joel Emer, and Vivienne Sze. 2016. 14.5 Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *Int'l Solid State Circuits Conf.* (2016).
[14] Steven Cooreman. 2016. Power-Saving Tips When Rapid Prototyping ARM Cortex-M MCUs. *http://electronicdesign.com/power/power-saving-tips-when-rapid-prototyping-arm-cortex-m-mcus* (2016).
[15] Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas Wenisch, and Ricardo Bianchini. 2012. CoScale: Coordinating CPU and Memory System DVFS in Server Sys. *Int'l Symp. on Microarch.* (2012).
[16] Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas Wenisch, and Ricardo Bianchini. 2012. MultiScale: Memory System DVFS with Multiple Memory Controllers. *Int'l Symp. on Low Power Elec. and Des.* (2012).
[17] Qingyuan Deng, David Meisner, Luis Ramos, Thomas Wenisch, and Ricardo Bianchini. 2011. MemScale: Active Low-Power Modes for Main Memory. *Int'l Conf. on Arch. Supp. for Prog. Lang. and Op. Sys.* (2011).

[18] Alessandro Stamatto Ferreira, Leonardo Cunha de Miranda, Erica Cunha de Miranda, and Sarah Gomes Sakamoto. 2013. A Survey of Interactive Sys. Based on Brain-Comp. Interfaces. *SBC Jnl. on 3D Interactive Sys.* 4, 1 (2013).

[19] Volker Gauck and Dieter Jaeger. 2000. The Control of Rate and Timing of Spikes in the Deep Cerebellar Nuclei by Inhibition. *Jnl. of Neuro.* 20, 8 (2000).

[20] Bernhard Graimann, Brendan Allison, and Gert Pfurtscheller. 2010. Brain-Comp. Interfaces: A Gentle Introduction. *The Frontiers Collection* (2010).

[21] Atif Hashmi, Hugues Berry, Olivier Temam, and Mikko Lipasti. 2008. Automatic Abstraction and Fault Tolerance in Cortical Microarch.s. *Int'l Symp. on Comp. Arch.* (2008).

[22] S Herbert and D Marculescu. 2007. Analysis of Dynamic Voltage/Frequency Scaling in Chip Multiprocessors. *Int'l Symp. on Low Power Elec. and Des.* (2007).

[23] Masayuki Hirata, Kojiro Matsushita, Takafumi Suzuki, Takeshi Yoshida, Fumihiro Sato, Shayne Morris, Takufumi Yanagisawa, Tetsu Goto, Mitsuo Kawato, and Toshiki Yoshimine. 2014. A Fully-Implantable Wireless System for Human Brain-Machine Interfaces Using Brain Surface Electrodes: W-HERBS. *IEEE Trans. on Comm.* E94-B, 9 (2014).

[24] Daniel Jimenez. 2003. Fast Path-Based Neural Branch Prediction. *Int'l Symp. on Comp. Arch.* (2003).

[25] Daniel Jimenez. 2005. Piecewise Linear Branch Prediction. *Int'l Symp. on Comp. Arch.* (2005).

[26] Daniel Jimenez and Calvin Lin. 2001. Dynamic Branch Prediction with Perceptrons. *Int'l Symp. on High Perf. Comp. Arch.* (2001).

[27] Daniel Jimenez and Calvin Lin. 2002. Neural Methods for Dynamic Branch Prediction. *ACM Trans. on Arch. and Code Optimization* (2002).

[28] Alok Joshi, Vahab Youssofzadeh, Vinith Vemana, T McGinnity, Girijesh Prasad, and KongFatt Wong-Lin. 2017. An Integrated Modelling Framework for Neural Circuits with Multiple Neuromodulators. *Jnl. of the Royal Soc. Interface* (2017).

[29] G Kaloshi, V Alikaj, A Rroji, G Vreto, and M Petrela. 2013. Visual and Auditory Hallucinations Revealing Cerebellar Extraventricular Neurocytoma: Uncommon Presentation for Uncommon Tumor in Uncommon Location. *General Hostpial Psychiatry* 35, 6 (2013).

[30] Stefanos Kaxiras and Margaret Martonosi. 2009. Comp. Arch. Techniques for Power Efficiency. *Synthesis Lecture Series* (2009).

[31] Ryan Kelly, Matthew Smith, Jason Samonds, Adam Kohn, A Bonds, J Movshon, and Tai Lee. 2007. Comparison of Recordings from Microelectrode Arrays and Single Electrodes in the Visual Cortex. *Jnl. of Neuro.* 27, 2 (2007).

[32] P Kohler, C Linsmeier, J Thelin, M Bengtsson, H Jorntell, M Garwicz, J Schouenborg, and L Wallman. 2009. Flexible Multi-Electrode Brain-Machine Interface for Recording in the Cerebellum. *IEEE Eng. Medicine Biology Soc.* 536, 8 (2009).

[33] Ki Yong Kwon, Seif Eldawatly, and Karim Oweiss. 2012. NeuroQuest: A Comprehensive Analysis Tool for Extracellular Neural Ensemble Recording. *Jnl. of Neuro. Methods* 204, 1 (2012).

[34] Charles Lefurgy, Kartik Rajamani, F Rawson, W Felter, M Kistler, and T Keller. 2003. Energy Management for Commercial Servers. *IEEE Comp.* 36, 12 (2003).

[35] Jian Li, Jose Martinez, and Michael Huang. 2004. The Thrifty Barrier: Energy-Aware Synchronization in Shared-Memory Multiprocessors. *Int'l Symp. on High Perf. Comp. Arch.* (2004).

[36] Sheng Li, Jung Ho Ahn, Richard Strong, Jay Brockman, Dean Tullsen, and Norman Jouppi. 2009. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Arch.s. *Int'l Symp. on Microarch.* (2009).

[37] Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Temam, Xiaobing Feng, Xuehai Zhou, and Yunji Chen. 2015. Pudiannao: A Polyvalent Machine Learning Accelerator. *Int'l Conf. on Arch. Supp. for Prog. Lang. and Op. Sys.* (2015).

[38] Xilin Liu, Basheer Subei, Milin Zhang, Andrew Richardson, Timothy Lucas, and Jan Van der Spiegel. 2014. The PennBMBI: A General Purpose Wireless Brain-Machine-Brain Interface System for Unrestrained Animals. *IEEE Int'l Symp. on Circuits and Sys.* (2014).

[39] N Matsumi, K Matsumoto, N Mishima, E Moriyama, T Furuta, A Nishimoto, and K Taguchi. 1993. Thermal Damage Threshold of Brain Tissue – Histological Study of Heated Normal Monkey Brains. *Neurol Med Chir* (1993).

[40] Scott McFarling. 1993. Combining Branch Predictors. *Tech. Rep. TN-36m, Digital Western Lab* (1993).

[41] David Meisner, Brian Gold, and Thomas Wenisch. 2009. PowerNap: Eliminating Server Idle Power. *Int'l Conf. on Arch. Supp. for Prog. Lang. and Op. Sys.* (2009).

[42] David Meisner, Christopher Sadler, Luis Barroso, W-D Webber, and Thomas Wenisch. 2011. Power Management of On-Line Data Intensive Services. *Int'l Symp. on Comp. Arch.* (2011).

[43] David Meisner and Thomas Wenisch. 2012. DreamWeaver: Arch. Supp. for Deep Sleep. *Int'l Conf. on Arch. Supp. for Prog. Lang. and Op. Sys.* (2012).

[44] Jorge Mercado, Javier Herrera, Arturo de Jesus Plansza, and Josefina Guiterrez. 2016. Embedded EEG Recording Module with Active Electrodes for Motor Imagery Brain-Comp. Interface. *IEEE Latin America Trans.* 75 (2016).

[45] Corinne Mestais, Guillaume Charvet, Fabien Sauter-Starace, Michael Foerster, David Ratel, and Alim Louis Bernabid. 2014. WIMAGINE: Wireless 64-Channel ECoG Recording Implant for Long Term Clinical Applications. *IEEE Trans. on Neural Sys. and Rehab. Eng.* 23, 1 (2014).

[46] Gregory Mone. 2015. Sens. on the Brain. *Comm. of the ACM* 60, 4 (2015).

[47] Christian Muhl, Brendan Allison, Anton Nijholt, and Guillaume Chanel. 2014. A Survey of Affective Brain Comp. Interfaces: Principles, State-of-the-Art, and Challenges. *Brain-Comp. Interfaces* 1, 2 (2014).

[48] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman Jouppi. 2007. CACTI 6.0: A Tool to Model Large Caches. *Int'l Symp. on Microarch.* (2007).

[49] Andrew Nere, Atif Hashmi, Mikko Lipasti, and Giulio Tononi. 2013. Bridging the Semantic Gap: Emulating Biological Neuronal Behavior with Simple Digital Neurons. *Int'l Symp. on High Perf. Comp. Arch.* (2013).

[50] TKT Nguyen, Zaneta Navratilova, Henrique Cabral, Ling Wang, Georges Gielen, FP Battaglia, and Carmen Bartic. 2014. Closed-Loop Optical Neural Stimulation Based on a 32-Channel Low-Noise Recording System with Online Spike Sorting. *Jnl. of Neural Eng.* 11, 4 (2014).

[51] C Nordhausen, E Maynard, and R Normann. 1996. Single Unit Recording Capabilities of a 100 Microelectrode Array. *Brain Res* (1996).

[52] Open Ephys Wiki. 2015. Possible Projects and Future Development. *https://open-ephys.atlassian.net/wiki/* (2015).

[53] Ilker Ozden, Megan Sullivan, Megan Lee, and Samuel Wang. 2009. Reliable Coding Emerges from Coactivation of Climbing Fibers in Microbands of Cerebellar Purkinje Neurons. *Jnl. of Neuro.* 29, 34 (2009).

[54] A Palumbo, F Amato, B Calabrese, M Cannataro, G Cocorullo, A Gambardella, P Guzzi, M Lanuzza, M Sturniolo, P Veltri, and P Vizza. 2015. An Embedded System for EEG Acquisition and Processing for Brain Comp. Interface Applications. *Wearable and Autonomous Bio. Dev. and Sys. for Smart Environments* 75 (2015).

[55] Dharmesh Parikh, Kevin Skadron, Yan Zhang, Marco Barcella, and Mircea Stan. 2002. Power Issues Related to Branch Prediction. *HPCA* (2002).

[56] Hernan Picard, Isabelle Amado, Sabine Mouchet-Mages, Jean-Pierre Olie, and Marie-Odile Krebs. 2008. The Role of the Cerebellum in Schizophrenia: An Update of Clinical, Cognitive, and Functional Evidences. *Schizo. Bull.* 34, 1 (2008).

[57] Steve Ramirez, Xu Liu, Pei-Ann Lin, Junghyup Suh, Michael Pignatelli, Roger Redondo, Tomas Ryan, and Susomo Tonegawa. 2013. Creating a False Memory in the Hippocampus. *Science* 341, 6144 (2013).

[58] Hernan Rey, Carlos Padreira, and Rodrigo Quiroga. 2015. Past, Present, and Future of Spike Sorting Techniques. *Elsevier Review* (2015).

[59] B Rosin, M Slovik, R Mitelman, M Rivlin-Etzion, S Haber, Z Israel, E Vaadia, and H Bergman. 2011. Closed-Loop Deep Brain Stimulation is Superior in Ameliorating Parkinsionism. *Neuron* 72, 2 (2011).

[60] James Smith. 1981. A Study of Branch Prediction Strategies. *Int'l Symp. on Comp. Arch.* (1981).

[61] James Smith. 2014. Efficient Digital Neurons for Larce Scale Cortical Arch.s. *Int'l Symp. on Comp. Arch.* (2014).

[62] ST Microelectronics. 2016. Ultra-low-power ARM Cortex-M4 32-bit datasheet. *http://www.st.com/content/ccc/resource/technical/document/datasheet/* (2016).

[63] Ian Stevenson and Konrad Kording. 2011. How Advances in Neural Recording Affect Data Analysis. *Nature Neuro.* 14 (2011).

[64] Yi Su, Sudhamayee Routhu, Kee Moon, Sung Lee, WooSub Youm, and Yusuf Ozturk. 2016. A Wireless 32-Channel Implantable Bidirectional Brain Machine Interface. *Sens.* 16, 1582 (2016).

[65] M Sullivan, A Nimmerjahn, D Sarkisov, F Helmchen, and SS-H Wang. 2005. In Vivo Calcium Imaging of Circuit Activity in Cerebellar Cortex. *Jnl. of Neurophys.* 94 (2005).

[66] S Suner, M Fellows, C Vargas-Irwin, G Nakata, and J Donoghue. 2005. Reliability of Signals From a Chronically Implanted, Silicon-Based Electrode Array in Non-Human Primate Primary Motor Cortex. *IEEE Trans. on Neural Sys. and Rehab. Eng.* (2005).

[67] D Tank, M Sugimori, J Connor, and R Llinas. 1998. Spatially Resolved Calcium Dynamics of Mammalian Purkinje Cells in Cerebellar Slice. *Science* 242 (1998).

[68] Elvira Teran, Zhe Wang, and Daniel Jimenez. 2016. Perceptron Learning for Reuse Prediction. *Int'l Symp. on Microarch.* (2016).

[69] Patrick Wolf. 2008. Thermal Considerations for the Des. of an Implanted Cortical Brain-Machine Interface (BMI). *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment* (2008).

[70] Pavel Yarmolenko, Eui Jung Moon, Chelsea Landon, Ashley Manzoor, Daryl Hochman, Benjamin Viglianti, and Mark Dewhirst. 2013. Thresholds for Thermal Damage to Normal Tissue: An Update. *Int'l Jnl. of Hyperthermia* (2013).

[71] Tse-Yu Yeh and Yale Patt. 1991. Two-Level Adaptive Training Branch Prediction. *Int'l Symp. on Microarch.* (1991).

[72] Stavros Zanos, Andrew Richardson, Larry Shupe, Frank Miles, and Eberhard Fetz. 2011. The Neurochip-2: An Autonomous Head-Fixed Comp. for Recording and Stimulating in Freely Behaving Monkeys. *IEEE Trans. on Neural Sys. and Rehab. Eng.* 19, 4 (2011).

[73] C De Zeeuw, J Simpson, C Hoogenraad, N Galjart, S Koekkoek, and T Ruigrok. 1998. Microcircuitry and the Function of the Inferior Olive. *Trends in Neuro.* 21, 9 (1998).