Energy-efficient Implantable Neural Decoder

Anil Bilgin, Cody Yang, Rebekah Zhao
Basic Asics
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania
Email: {abilgin,cyyang,biqiz}@andrew.cmu.edu
March 8 2018

***Abstract*—We propose to design a machine learning based neural decoding chip with implantability and low power goals in mind. Neural decoders are essential to brain machine interfaces and implantable ones reduce risk of infection as well as increase portability. Our design will be simulated both in MATLAB and in Verilog-AMS and implemented using a 45nm CMOS process in Cadence.**

***Keywords*—Neural Decoder, Energy-Efficient, Extreme Learning Machine, Current Mirror Array**

I. Project Description

Brain-machine interfaces have become an increasingly popular topic over the past decade, facilitating the study of neural prosthetics to assist disabled patients. In implantable neural devices, electrodes are placed inside the brain to record neural signals. Action potentials generated by neurons in the brain are detected and converted into spike trains. Neural decoding is performed subsequently to map spike train patterns to motor intentions of the brain. State-of-the art neural decoding generally relies on the neural implant to wirelessly transmit data out of the brain and process data on PC. However, the resulting power consumption is huge, and wireless transmitting data requires signals to be samples, resulting in loss of data as well.

In order to reduce power consumption and data loss, in this project we will be designing a fully implantable neural decoder with the architecture proposed by Chen et al [1]. In the algorithm named Extreme Learning Machine (ELM) [2]-[3], there is only one hidden layer. The advantageous part of this algorithm is that it doesn't require the inputs weights to be trained, therefore these weights can be randomized and not touched again. These input weights to this hidden layer is implemented with fabrication mismatches in a current mirror array to reduce power consumption. Therefore, the full system consists of a DSP processor performing spike detection and sorting and a Machine Learning Coprocessor (MLCP) containing current mirror arrays to generate the input weights. The hidden layer output is then fed back to the DSP processor to produce the output. For demonstration purposes, we will only implement the MLCP in 45nm GPDK. Inputs to the MLCP are spike trains recorded from a monkey's premotor cortex. Outputs from the MLCP will be extracted and processed in software.

II. Design Requirements

A. Decoding Accuracy

Our first design requirement is decoding accuracy of our machine learning algorithm, the ELM, for the specific application of decoding direction of cursor movements controlled by a monkey. Chen et al [1] achieved 99.3% decoding accuracy for detecting monkey's finger movement types. We're aiming for the same decoding accuracy, since we know that this value is achievable.

B. Power Consumption

The second design requirement is that our chip implementation consumes low power. Chen et al [1] managed to have a total of 414 nW power consumption in their 128 channel implementation of the MLCP (which doesn't include the DSP). This gives a power consumption per channel of 3.2 nW's. We looked into the average firing rate of our neural data and found around 1.4 firings for every 100 milliseconds. Considering the output of our DAC will be in the range of 2-126 nA's (explained later) and a VDD = 1V, we can estimate the average power consumption per current mirror array to be 2.8 mW. This means that we can estimate that we will have L x 2.8 nW power consumption per channel, where L is the dimension of our hidden layer. L is going to be tuned as a hyperparameter in our design to give us the best possible decoding accuracy.

III. Functional Architecture

A. Software Components

In a complete and readily implantable device, the neural data would come from an electrode array sensor and be received by the DSP, which performs preprocessing (eg. spike detection, spike sorting, signal amplification, noise filtering, etc) but for our purposes we have acquired an open-source dataset to preprocess the data in MATLAB instead. We then import the data into a Verilog testbench, as well as generate the control signals needed by the chip (eg. clocking, reset, delay selection, and signals for reducing power in components).

B. Components

- Window Counter: The window counter converts the spike train input into a digital number representing the frequency of spikes in each channel.

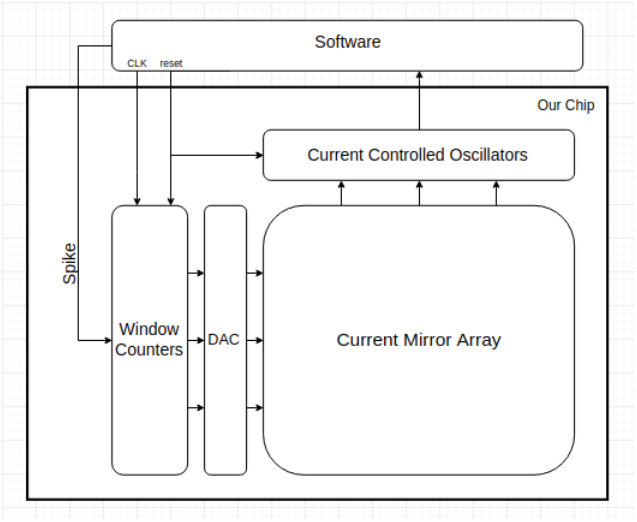

Fig. 1. Block Diagram

- Digital to Analog Converter: The digital to analog converter (DAC) takes a digital input from the window counter and outputs an analog current.
- Current Mirror Array: The current mirror array generates random weights through device mismatch during the fabrication process. The mirroring and summing of currents performs the dot product of input vector with random weights to compute the hidden layer nodes' inputs.
- Current Controlled Oscillator: The current controlled oscillator (CCO) receives summed currents from the current mirror array and outputs a pulse frequency modulated signal with the frequency proportional to the total current input.

IV. Design Trade Studies

In this section, we will discuss the detailed design requirements and testing plans for each above-mentioned components.

A. Window Counter

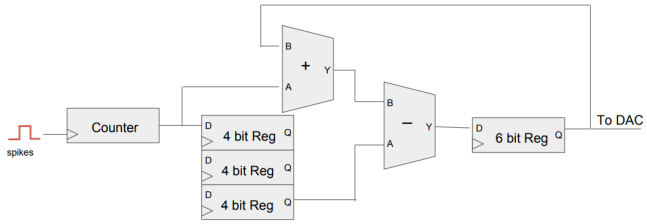

Fig. 2. Block Diagram for Window Counter

The window counter converts the spike train signal into spike frequency values for the input vectors to the ELM algorithm. The counter portion consists of an adder and a register. Since the frequency of spikes is relatively slow, a simple ripple-carry adder topology will suffice. With window sizes of 20ms and maximum spike frequency of 600Hz, the maximum number of spikes counted within one window is 12 spikes which corresponds to a 4-bit counter.

The rest of the window counter consists of a chain of registers for storing most recent spike counts, two adders to perform addition of new spike counts and subtraction of old counts. Finally there is a 6-bit register for storing the current window counter value. The main purpose of this part of the window counter is to have a moving window average and gain more information about the most recently fired neurons by representing them as features in our input vector x.

B. DAC

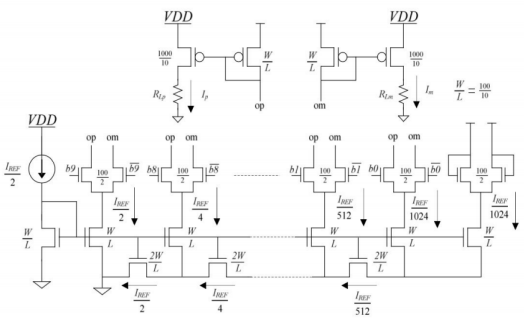

Fig. 3. Circuit Schematic for DAC

The main function of the DAC is to convert the output count from the window counter into analog currents. Since the window counters have a 6-bit resolution, the DAC is also 6-bit wide. As shown in the figure above, the DAC uses a pure CMOS W-2W topology. The load to each DAC is a single current mirror array. To guarantee that the output of each DAC is reliable enough, the most important testing factor is differential nonlinearity (DNL). We would like to achieve a DNL of within ± 3 bits.

C. Current mirror array

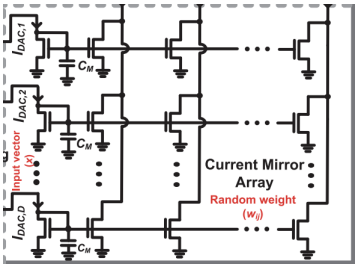

Fig. 4. Circuit Schematic for Current Mirror Arrays

The current mirror arrays generate the random weights from the input layer to the hidden layer through device mismatch during the fabrication process. Each column of the arrays applies random weights to the input vector, effectively implementing dot product computation of each hidden layer node's input. The size of this matrix of current mirrors will be determined by hyperparameters (namely the number of hidden layer nodes) that maximizes the algorithm's performance.

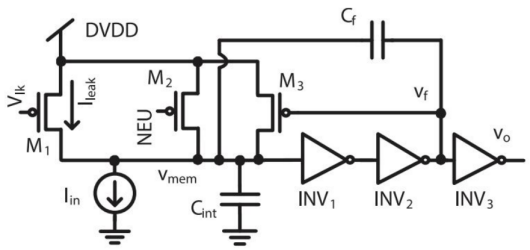

Fig. 5. Circuit Schematic for Current Controlled Oscillators

D. CCO

The CCO converts outputs of summed currents into a frequency modulated signal with the frequency proportional to input current. A counter is connected at the output of the CCO to count the number of pulses. This way the current is converted to digitally readable information. Implementation of CCO is shown in the above figure. The capacitance of $C_i nt$ sets the oscillation frequency. C_f provides hysteresis through positive feedback to maintain oscillation of circuit [1]. To test this circuit, we will input spike trains of frequency ranging from 0 to 630 Hz. By taking the variance of the counter measured for the same inputs, we can evaluate the jitter performance. Because Chen et al [1] achieved jitter less than 0.1%, this will be our initial target.

E. Software Algorithm Implementation

The software parts of our project is only at the very beginning and at the very end. Our data is processed in MATLAB and turned into proper spike trains for the input of our circuit, which is going to be copied as a digital signal in Verilog at the input stage to the window counters. In addition to that, MATLAB code is going to be necessary to analyze and train the weights (from the hidden layer to the output layer). Our simulation of the CCO and the counters (on top of the CCO's) in Cadence is going to dump out files, which we will then parse with MATLAB and proceed to use the rest of the ELM algorithm to train our model and get our decoding accuracy.

V. Design Work Flow

There are four phases of our design process. These include designing specifications, writing verilog-AMS to test and finalize specifications, replacing individual blocks with Cadence circuit diagram, and eventually laying out the entire circuit.

- Design Specifications: in this phase, we started with choosing the circuit topology of entire chip and then chose the circuit topology for each component. After simulating each circuit in Cadence with proper sizing, we were able to understand the input and output range of each component. This way we were able to decide the specifications for the whole system.
- After deciding specifications, we will write verilog-AMS to simulate them. Through testing we can evaluate if they meet our targets. If not we will adjust specifications until eventually all targets are met.
- After finalizing design specifications, we will start designing circuits in Cadence block by block and incorporate them with verilog-AMS. When all blocks are implemented in circuit, we test the final performance of circuit.
- Finally, we layout the schematic we designed to make our project a manufacturable chip. Even though the 45nm GPDK process is not actually manufacturable, we would still make sure the chip is ready for fabrication.

VI. Project Management

A. Schedule

We divided up our work based on each team member's individual confidence in the given area, and tried to come up with a timeline that's shown in Figure 3. We're mostly going to be collaborating on everything together towards the latter part of the project.

Week	Anil	Rebekah	Cody
2/19		Study and implement CCO	Work on the window counter
2/26	Process dataset into usable format	IO & specs of each component (eg current, noise tolerance, control signals, clocking)	
3/5	Hyperparameter optimization		
3/12	Write verilog-AMS(DAC, window counter, current mirror, CCO, counter)		
3/19	Spring Break!		
3/26	Continue to write Verilog		
4/2	Test our verilog design		
4/9	Build circuit (current mirror)	Build circuit (DAC, CCO)	Build circuit (adder, flip-flops)
4/16	Carnival!		
4/23	Put circuit together and test		
4/30			
5/1	Layout		
5/7	Demo		

Fig. 6. Our schedule of the work load in between weeks.

B. Team Member Responsibilities

- Anil is going to be mainly responsible for the ELM algorithm, how it works, its optimization using hyper-parameters and performance metrics using the hidden node outputs from the MLCP. He's also going to be responsible for implementing the current mirror array in Cadence as well as contributing to Rebekah and Cody's efforts in building other Verilog components in digital and parts such as the DAC and CCO in Cadence. Towards the end of the project, everyone is going to collaboratively work on putting the circuit together and layout.
- Rebekah is going to be responsible in determining design specs for most of the analog parts in this chip. She's primarily responsible for building of the DAC and the CCO as well as helping Cody and Anil testing other Verilog components and simulating circuits in Cadence. Towards the end of the project, everyone is going to collaboratively work on putting the circuit together and layout.
- Cody is mainly responsible for designing and building most of the digital parts in this chip. He's primarily responsible for designing and building the window

counter and the counters after the CCO. He's going to write Verilog-AMS to simulate certain circuit blocks and finally going to collaboratively work on putting the circuit together and laying it out.

C. Budget

Luckily this project doesn't need any budget since it's fully a simulation based project. We don't need to buy any components since all of the design software we are going to use is readily available to us.

D. Risk Management

There are several risk factors involved in this project, and they can be summarized to:

- We may or may not be able to achieve the target decoding accuracy of 99.3% in our design, simply because we're using a different dataset than Chen et al [1] and our dataset may not be as classifiable as theirs. This is why we're looking into two different sets of datasets at the same time, and trying to optimize both of them. Decoding accuracy is also highly dependent on hyperparameter optimization, which is also what we're currently working on in order to determine our hyperparameters early on.
- Another concern is time management to complete this project. Certain parts of the project may end up taking a lot more time than we assigned them to in our weekly schedule. We're working as hard as we can in order to keep up with this schedule, but there's a possibility that we may not be able to overcome a certain part of the project that we're not very familiar with in its reserved time frame. If this seems to be the case, we're thinking of getting ideas about how to approach the problem from our TA's, professors and other senior graduate students who have more expertise in this field.

VII. Related Work

While coming up with design ideas, we came across several papers targeting similar functions.

- Rapoport et al: A Biomimetic Adaptive Algorithm and Low-Power Architecture for Implantable Neural Decoders [4]. This algorithm implements a continuous-time artificial neural network with a bank of adaptive linear filters using analog computing. However, no measurement results are published to support the silicon viability of the architecture [1].
- Rapoport et al: Efficient Universal Computing Architectures for Decoding Neural Activity [5]. This paper proposes a very simple algorithm suitable for highly scalable implantable systems as shown in the figure below. The architecture implements a universal computing machine emulating the dynamics of a network of integrate-and-fire neurons, and requires only counting. Neural decoding is done using only logic operations. This algorithm was then implemented on an FPGA and consumes 537 μW.

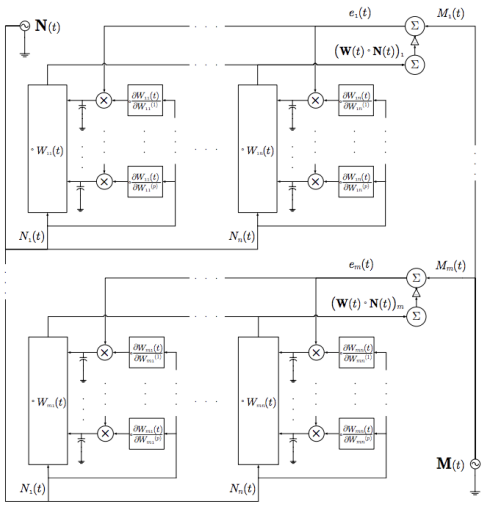

Fig. 7. Circuit Diagram of Algorithm in [4]

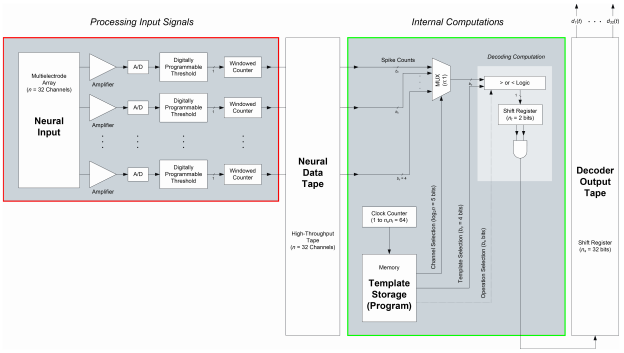

Fig. 8. Circuit Diagram of Algorithm in [5]

- Qiao et al: A reconfigurable online-learning spiking neuromorphic processor comprising 256 neurons and 128K synapses [6]. This paper implemented a full-custom mixed-signal VLSI device with neuromorphic learning circuits that emulate the biophysics of real spiking neurons and dynamic synapses for exploring the properties of computational neuroscience models.This chip was fabricated on 180nm CMOS process and occupies an area of $51.4mm^2$. It consumes a total of 4 mW for typical experiments.

Compared to the above implementations, our design has several advantages. First, it is more power efficient. The power consumption for the MLCP is in the nano Watt range. Secondly, the algorithm the MLCP implements is very simple compared to that of [6]. Lastly, unlike [4], this algorithm has been implemented and tested before, which makes it a more reliable approach.

References

[1] Y. Chen, E. Yao and A. Basu, "A 128-Channel Extreme Learning Machine-Based Neural Decoder for Brain Machine Interfaces," in IEEE Transactions on Biomedical Circuits and Systems, vol. 10, no. 3, pp. 679-692, June 2016.

[2] G. B. Huang, Q. Y. Zhu, and C. K. Siew, Extreme learning machines: Theory and applications, Neurocomputing, vol. 70, pp. 489501, 2006.

[3] G. Huang, H. Zhou, X. Ding, and R. Zhang, Extreme learning ma- chine for regression and multiclass classification, IEEE Trans. Sys- tems, Man, Cybern. B, Cybern., vol. 42, no. 2, pp. 513529, Apr. 2012.

[4] B. Rapoport, W. Wattanapanitch, H. Penagos, S. Musallam, R. Andersen, and R. Sarpeshkar, A biomimetic adaptive algorithm and low-power architecture for implantable neural decoders, in Proc. 31st Annu. Int. Conf. IEEE EMBS, 2009.

[5] Qiao, Ning et al. A Reconfigurable on-Line Learning Spiking Neuromorphic Processor Comprising 256 Neurons and 128K Synapses. Frontiers in Neuroscience 9 (2015): 141. PMC. Web. 9 Mar. 2018.