# Energy Efficient On-Chip Neural Feature Extraction for Brain-Computer-Interfaces

Thesis by
Steven Patrick Bulfer

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy


CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended September 5th 2025

Steven Patrick Bulfer
ORCID: 0000-0001-9942-1195

*To my wonderful parents, Joyce and Patrick,*
*For your support, love, sacrifice, and belief in me.*
*To my Aunt Mary,*
*For your constant care packages and messages that cheer me on.*
*To my friends and loved ones,*
*For helping me up when I get knocked down and keeping me laughing.*

# ACKNOWLEDGEMENTS

My journey at Caltech has shaped me personally in ways I never even imagined possible. Each trial and tribulation presented an opportunity for growth made possible by the immense academic generosity of the Caltech community. My research experience showed me who I am and what I am capable of. The support of my wonderful friends, family, and advisor gave me the strength to never give up.

I am deeply grateful to my advisor, Professor Azita Emami, for pushing me to do my best in every aspect of my work. From early in my career, I came to understand how easy it is to become overwhelmed and adrift in the sea of human knowledge. She showed me the importance of breaking down problems into tractable pieces and testing them rigorously. She is also someone who I know cares deeply for her students, not just academically but personally, teaching me the importance of patience and empathy in research and mentorship.

I am very thankful for the opportunity to work with my collaborators, Professor Richard A. Anderson and his lab, including Dr. Tyson Afflalo, Dr. Spencer Kellis, Dr. Abraham Betancourt Vera, and Nikos Mynhier. He and his lab gave me an abundance of help understanding the neuroscience behind brain-machine interfaces, and provided resources in every necessary aspect to help me complete my work. I would like to especially thank Dr. Jorge Gámez de Leon for his unwavering support and advice. He helped me through some of the most difficult hurdles in my Ph.D. and I will forever be thankful for his mentorship. I would also like to thank JJ for his participation in our experiments, persistent faith in our success, and infectious optimism. His enthusiasm for our work is truly what makes it all worthwhile.

I am very honored to have Professor Volnei Pedroni, Professor Alireza Marandi, and Professor Ali Hajimiri as members of my candidacy and defense committee. Professor Volnei Pedroni helped me refine my scientific communication skills, and provided invaluable technical direction for digital architecture design. He helped me organize ideas stuck in the ether of my mind into concrete design descriptions, which is essential for scientific communication and the interrogation of those ideas for improvement. Both Professor Ali Hajimiri and Alireza Marandi provided essen-

tial support in helping me navigate the trials of a Ph.D. and I will be forever grateful for their advice. I have always felt welcomed by those from whom I seek help, which really accentuates what a privilege it has been to be a part of this community.

It is hard to put into words how much the people of the MICS lab mean to me. I truly could not have done this without them. It has been such a honor to work with and befriend Dr. Arian Hashemi, Dr. Fatemeh Aghlmand, Dr. Saransh Sharma, Dr. Benyamin Haghi, Lin Ma, Shawn Sheng, Ting-Yu Cheng, Hayward Melton, Dr. Ziyi Chang, Dr. Loai Danial, Mohamadamin Panahandeh, Shengsheng Wang, Johan Razavi, and Mohammaderfan Ramesh, and everyone else who has touched this lab. Their community, support, and scientific expertise shaped me as much as a scientist as it did as a person. I would also like to thank Michelle Chen for her constant vigilance to make this lab run smoothly.

Last, but certainly not least, I would like to thank the broader Caltech community and the people I met along my journey. I would like to give a special thanks to Mauro Ferreira Santos for his personal mentorship and guidance. The most important thing I learned during my time here is that despite everything that happens around you, or to you, community is how we endure. Science is about people as much as it is about knowledge, and what makes Caltech the success that it is, are the wonderful people who constitute its community. I have no doubt that despite the uncertainty of what is to come, Caltech will endure not because of its resources or accolades, but because of its people and what they care about: truth, integrity, and each other.

# ABSTRACT

Neural interfaces are entering an era where what once was science fiction is becoming a reality. As neural interfaces move out of the lab and into people's lives, the stability of neural decoding algorithms becomes ever more pressing. It is an unfortunate reality that neural implants degrade from long-term exposure to the neurological environment, however prior work has shown enhanced decoding stability in the application of 1D convolutional neural networks to neural feature extraction. However, these algorithms have high memory and processing requirements, prohibiting them from meeting the low area and power restrictions of implantable brain-machine interface decoding pipelines.

This dissertation addresses the difficulties of implementing these algorithms on streamed neural data with high parallelism and low area and power costs. We address the unique dataflow characteristics of the feature extraction workload by designing a tailored processing element that reduces the memory access requirements by 2×. We further reduce system memory requirements through efficient process scheduling and memory partitioning. We then address the model complexity through retraining and analysis of the effect of various system parameters on the accuracy of kinematic decoding and hardware performance.

Results show that these design choices were able to successfully implement these intensive but performant algorithms within the power and area budgets of implantable devices. The architecture supports 192 channels that achieve state-of-the-art decoding stability at 1.8 $\mu W$ and 12801 $\mu m^2$ per channel in 65 nm CMOS technology. The device is a fully configurable, scalable, area and power efficient solution that supports models with 2-8 feature layers and a total kernel length of up to 256. This architecture reduces caching requirements by 5× over conventional computation schemes. We show our hardware optimized models maintain superior stability over time using recorded data from tetraplegic human participants with spinal cord injury. The models and hardware were validated in real time with a human subject in online closed-loop center-out cursor control experiments with micro-electrode arrays that were implanted for 6 years. Decoders using features generated with this work substantially improve the viability of long-term neural implants compared to other feature extraction methods currently present in low-power BMI hardware.

# PUBLISHED CONTENT AND CONTRIBUTIONS

S. Bulfer, J. Gámez, A. Yan-Huang, B. Haghi, V. Pedroni, R. A. Andersen, A. Emami "**A 192-Channel 1D CNN-Based Neural Feature Extractor in 65nm CMOS for Brain-Machine Interfaces**", In: *IEEE Transactions on Biomedical Circuits and Systems* (In Revision)
SB participated in conceiving the ideas and was lead designer of the CMOS chip, optimized FENet models for implementation, built the validation system and helped perform online experiments, analyzed results, and wrote the manuscript.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

*Chapter 1*

# INTRODUCTION

As we enter this era of neuroscience and engineering, our ever-curious minds have begun to not only probe the inner workings of our own thoughts, but also endeavor to communicate with the very mechanisms from which these thoughts are derived. Neural electronic interfaces promise insight and control over the symphonic chaos of such biological neural circuits. In the past few decades, successes in neural engineering have allowed those afflicted with serious spinal cord injuries new avenues to regain control over the physical world by decoding kinematic intent from the cortex directly [39]. Devices that interpret the complex encoding of the human brain into information that computers and actuators understand are colloquially known as brain-machine interfaces.

## 1.1 Brain-Machine Interfaces

A brain-machine interface is our endeavor as scientists and engineers to bypass the frailty of biological periphery. This includes any device that attempts to decode an underlying neural state by measuring signals directly from the brain. Ascertaining internal neural states allows automated systems for both the treatment, and mitigation of neurological ailments such as paralysis, seizure disorders, Parkinson's, Alzheimer's, and depression. As we develop higher-fidelity decoding systems and integrate AI into their decoding pipelines, even more complex interfacing systems are becoming possible such as direct speech decoding [16] and neuro-rehabilitation, where damaged neural circuits are retrained using closed loop stimulation [29]. The decoding of neural activity has been utilized to control a wide breadth of actuators, including wheelchairs [43, 38], drones [28], robotic limbs [10], digital communication systems [6, 1], and computer cursors [10, 48, 31, 62, 20]. These interfaces have also been coupled to muscle stimulation devices, allowing patients to once again control their own muscles [5]. The general components of a brain-machine interface system are shown in Figure 1.1 A.

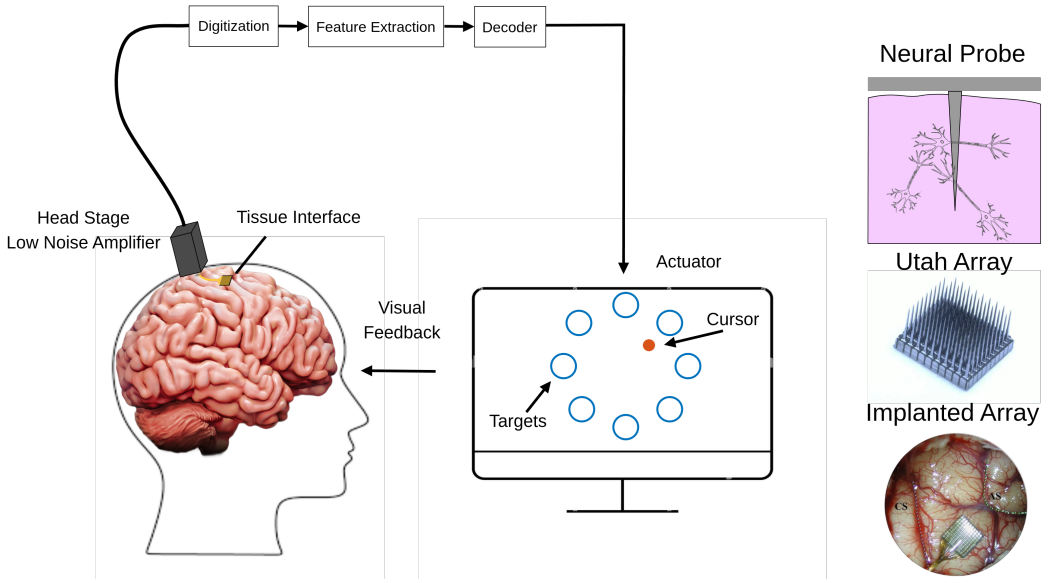



Figure 1.1: General components of a brain-machine interface: (A) High-level system schematic. (B) Components of a tissue interface.

A brain machine interface system is broadly composed of a tissue interface, amplifier, digitization system, feature extraction, decoding, and actuators. The neural tissue interface is composed of electrodes, antennas, or ultrasonic transducers which are the first step in transforming or transporting some indicator of neural activity into signals to be processed. Within this work, we focus on recordings from a type of multi-electrode array known as a Utah array as depicted in Figure 1.1 B. This tissue interface is designed to physically penetrate the neural tissue to record the local electrical environment close to live neurons. Once the electrical signals are sensed by the probes, they must be amplified and digitized for further processing to concentrate information through various methods of filtering and feature extraction. Decoding is performed on these concentrated metrics by relating where neural activity is recorded to neural states of the brain. These states can then be used to interface a person's brain with a plethora of physical and virtual actuators that the implanted person can interact with using only their mind.

Brain-machine interfaces experience a tradeoff between the resolution of the measurement [30], and the invasive nature of such systems, as shown in Figure 1.2. The finer the resolution of neural activity that is required for a particular recording system, the more intrusive recording equipment needs to be in order to distinguish activity. On the least invasive end of the spectrum, Electroencephalograms (EEG), can discern only broad strokes of brain activity from ultra-weak remnants of electric field that escape the skull. As of now, the highest resolution neural probes require direct contact with the region of the brain of interest.

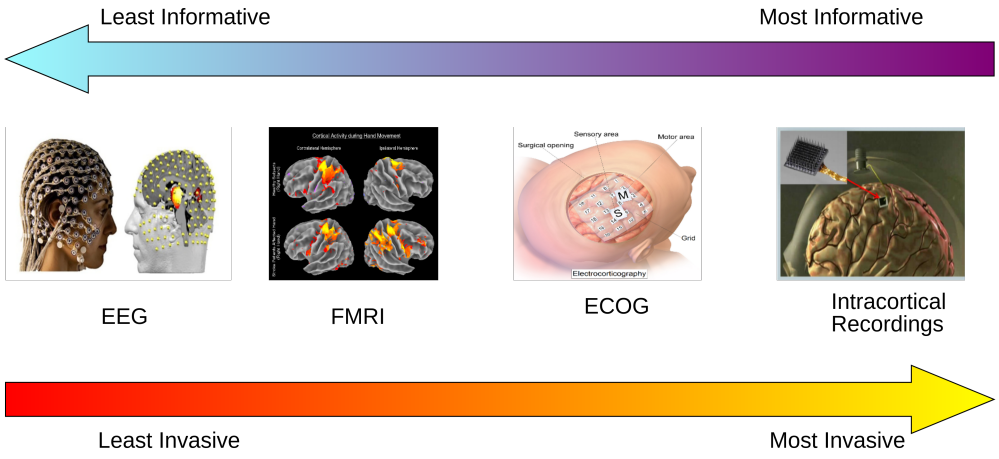



Figure 1.2: An overview of the spectrum of brain-machine interfaces.

Neural implants face a challenge of high processing demand, small power and area budgets, nonstationary neuro-physiological dynamics, high noise, and biological environments that are corrosive to electrodes [55, 50, 53, 4, 47, 26]. Area budgets are constrained by the necessity to share physical die space with analog front ends which capture, filter, and digitize neural streams. The ability to co-locate digital processors with digitizers reduces power by minimizing the capacitance signals must travel through between data generators and processors. Deep learning algorithms designed to improve neural decoding performance, however, have large memory and algorithmic complexity, posing significant challenges to their integration within the neural decoding environment.

## 1.2 Streaming Processors

FENet [22] is a deep neural network algorithm which has been shown to be an excellent neural data transformer which provides improved performance over non-tailored transforms. Designing an efficient architecture for deep neural network processors with application to neural interfaces first requires an understanding of the unique form in which neural data is received. Neural data are streamed from recording front ends, meaning that each sample is received in sequential order. However, neural data transformers operate on bins of data, which require a history of neural recordings to be stored during computation. Most deep neural network hardware architectures are designed for batch processing. Batch processing can take advantage of spatially unrolling computations and activation reuse. While these systems can have very high computational efficiencies since processing elements can locally pass activations between each other, the memory required to cache all of the activations before starting the computation quickly degrades any power reduction made during computation. Streaming processors, on the other hand, tailor their execution model to ameliorate the difficulties of processing temporally structured data. An abstract example of the differences between batch and stream processing is shown in Figure 1.3.

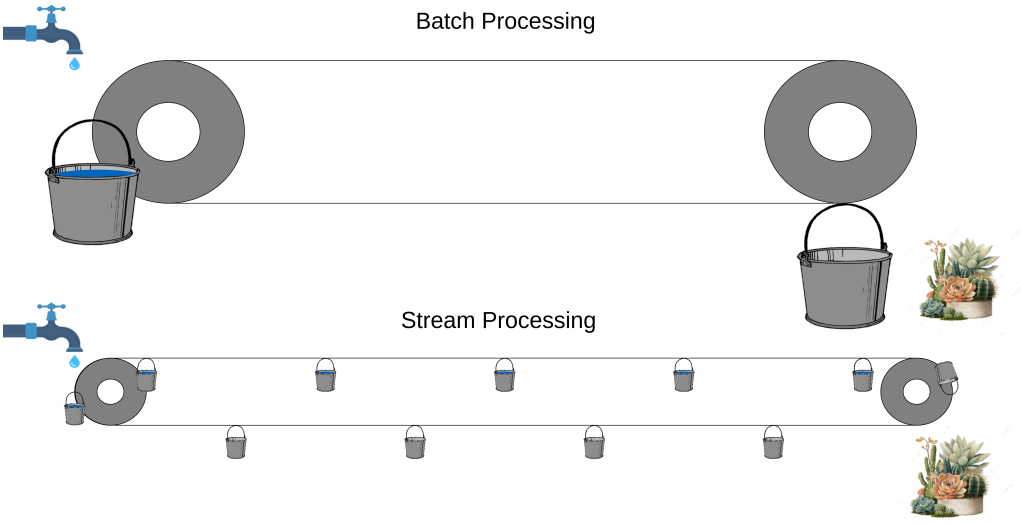



Figure 1.3: Representation of the difference between processing in batches vs. streams.

In this toy example, the object of the system is to relay water from a source (the leaky faucet), to its destination (a classic southern California garden). If the source were unrestricted, the most efficient solution for transporting water from one destination to the other would be to use a single large bucket to make as few trips as possible. Unfortunately, due to drought, the water source is constrained on the rate at which it can provide water. In this case, batch processing represents the system with a large bucket. This system requires heavy duty conveyor cables and a larger drive motor to provide higher impulse power, which all sit idle while the bucket spends time filling with water. A streaming processor, on the other hand, saves on materials and power by closely matching the system requirements to the temporal nature of its workload.

Overall, streaming processors trade generality for efficiency: by aligning the hardware's execution model with the temporal structure of the data, they deliver high throughput per watt and predictable latency. These properties are essential for processing deep neural networks at the edge and in real-time neural systems.

### 1.3 Contribution

In this dissertation, I present novelty to neural feature extraction hardware and CNN accelerators through hardware-software co-design and optimization to implement the FENet algorithm at low area and power. This is accomplished in part through an efficient task scheduler for 1D convolutional layers operating on streamed inputs, with flexibility to implement a breadth of filter lengths and strides. The scheduling algorithm allows direct reduction of the number of required memory elements from O(L log[N-1]), to O(L) where N is the length of samples in the input data bin and L is the sum total of kernel lengths of all layers. This architecture combines stride-aware streaming and control sequencing that tailors this dataflow to the FENet workload specifically for high channel counts using minimal memory resources.

A novel processing element is designed to ameliorate the requirements of the FENet dataflow. The element contains dual convolution paths on each neural data stream with multiple output modes, one path that generates the output neural features and another that generates activations for subsequent layers. Other CNN architectures are not designed for this multi-modal dataflow and would require two separate read-processes-write operations for each channel. The wide bandwidth data network is optimized for the high-channel count nature of neural signal processing, maximizing processing element utilization.

To minimize area requirements for each channel while taking advantage of the relaxed timing constraints gained by processing neural data sequentially, the processing hardware is minimized to use a single 8-bit adder and two 16-bit barrel shifters to compute all of the operations of the FENet workload by subdividing each operation into simpler shift-and-add steps. Furthermore, word-serial processing allows controls to be gated based on individual weight bits of kernels; taking advantage of the bit-sparsity of the FENet weights. This feature extraction system has a wide bandwidth data write bus that enables simultaneous streaming of data writes from individual sources. To minimize leakage power, which can become significant in highly optimized feature extraction tasks, channel-and-layer-level power gating is used over the entire device. This architecture is a first of its kind for implementing the FENet workload within the tight power and area constraints of implantable neural devices.

We fabricated this architecture in 65 nm CMOS and fully verified the system and hardware optimized FENet models through offline validation and closed-loop kinematic decoding with a human subject. We showed efficient feature extraction that enabled accurate cursor control in a patient with an implant that has been in place for 6 years. These features allow high-quality kinematic decoding after traditional feature extraction hardware has lost viability.

### 1.4 Organization

This thesis is organized by the various disciplines covered throughout the hardware-software co-design of an architecture which can efficiently implement the FENet workload. Chapter 2 begins the journey by reviewing the various methods of neural feature extraction, and prior architectures developed to implement them. These methods have their uses within the neural decoding environment, however they fall short when decoding kinematic intent from chronically implanted neural probes. We introduce prior work utilizing the convolutional neural network model 'FENet' to address those deficiencies in chronic implant decoding stability. Following the review of this algorithmic solution, we take a look at the various techniques in present literature for hardware implementation of convolutional neural networks, and assess their viability for the FENet workload. Through these explorations, we uncover what abstract architectural design requirements are necessary for efficient implementation.

Chapter 3 provides an overview of the hardware architecture and design methodology for major components of the CMOS chip with a special focus on the custom design

of the processing element. We assess the viability of various design approaches and determine the necessity of safeguards such as overflow clamping, granular power gating, and zero padding. We introduce the convolution scheduling algorithm designed to optimize memory resources, and a zero-padding scheme which avoids unnecessary MAC operations for the padding-heavy FENet workload. The time-domain-multiplexed computational tasks and their behavior are described to fully illustrate the inner workings of the processing element. The parameters of the FENet model and their effect on both the performance, and hardware requirements are analyzed in Chapter 4. Using this analysis, the model is optimized to reduce the computational complexity on the hardware implementation. Finally, these models are validated and compared with other feature extraction techniques implemented in hardware using neural recording data. The hardware is then fully validated online with a human participant in loop with the feature extraction system.

The architecture is fabricated in 65 nm CMOS and measured in Chapter 5. Both the static and dynamic power requirements of the system are characterized with various configurations of the system to build a power model for power-performance optimization. The effectiveness of implementing the FENet algorithm within the constraints of the neural decoding environment is analyzed with regards to the area costs, power efficiency, and scalability to even higher channel-count BMIs. The VDD sensitivity of the chip is assessed and potential performance improvements are suggested. Finally, this feature extraction ASIC is compared with other state-of-the-art neural feature extraction hardware. Chapter 6 concludes this work, and provides a number of future directions to improve the architecture for area and power, as well as mitigate undesirable behavior.

*Chapter 2*

# BACKGROUND AND PRIOR ART

This chapter will provide an introduction to the various fields that are vital to the implementation of brain-machine interfaces in hardware. We will begin with an introduction into the way neural information is represented in the brain and various techniques that are used to enhance and extract the presence of this information in intracortical recordings. However, these methods struggle to decode kinematics from probes that have been implanted for long periods of time. Chronic exposure to the neural environment causes degradation of neural probes and scarring of neural tissue, leading to a significant decrease in neural signal-to-noise ratios. To address these noise issues, a data-driven algorithm called FENet was designed to improve kinematic control through the use of a convolutional neural network. Once we are familiar with the decoding problem and the algorithm we are using to solve it, we will then introduce ourselves to the various processor architectures and techniques used for edge-ML processing so that we understand the design framework necessary to implement this algorithm at low area and power, suitable for implantable devices. Using this framework, the hardware architecture will be custom-tailored to efficiently implement the FENet workload.

## 2.1 Neural Information

To understand how to extract information from neural data, we must first understand how neural activity is biologically encoded. All neural activity, from the simplest neural circuits found in nematodes, to the stupendous complexity of thought capable of the human brain, is mediated through state transfers between neurons, referred to as *neural spikes*. The diagram in Figure 2.1 depicts the cycle of a neuron's firing sequence. The complex behavior of a neuron is driven by ion concentration gradients across its cell membrane. Sodium and potassium ion pumps embedded in the membrane spend energy to sustain these gradients, inducing both an electrical and chemical potential in steady state. A resting neuron will drive and excess of sodium and a deficiency of potassium inside the cell. Due to the difference in pump energy, there is approximately a -70 mV voltage potential that is built during cell quiescence. This resting membrane potential is modulated by signals received from neighboring neurons through junctions called synapses.

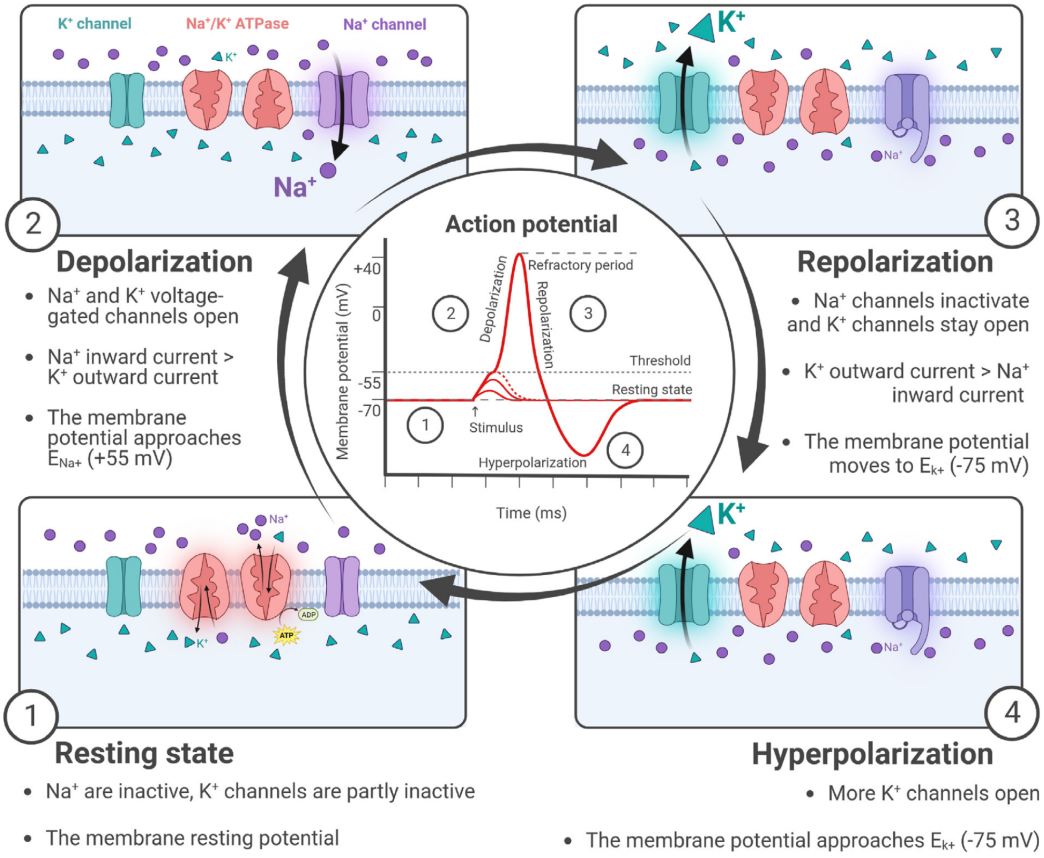



Figure 2.1: Depiction of the phase of a neural spike.
1) Ion pumps maintain membrane voltage and ion gradients to their resting potentials. 2) Initial stimulus initiates a perturbation of the neural membrane potential. A threshold is reached that initiates the opening of sodium channels causing depolarization of the membrane. 3) Sodium channels deactivate and the slower calcium channels open, repolarizing the membrane potential. 4) The calcium pumps overcorrect the membrane voltage, putting it in a hyper-polarized state[36].

These synapses are the fundamental computational unit of biological neural circuits because that information exchange can be gated through complex chemical processes which tune their coupling strength between neurons. The activation of a synapse will either increase or decrease the cell membrane potential depending on each synapse's chemical tuning. These gated connections allow for a plethora of complex emergent behavior in biology that we are still trying to fully understand.

The primary enabler to a spiking event are the voltage sensitive ion gates which lie within a neuron's cell membrane. Designed through millions of years of evolution, these ion gates react to the current state of the cell membrane's voltage potential. As depicted in Figure 2.1, a cell membrane voltage threshold crossing event will first trigger the sodium gates, allowing the release of the stored chemical gradient of sodium ions. The inrush of sodium depolarizes the cell membrane to become significantly more positive than the resting membrane potential, resulting in the activation of the potassium gates; quickly counteracting the flood of sodium ions to re-polarize and reset the system. As the sodium pumps deactivate and the potassium pumps work to place the membrane potential back to equilibrium, the neuron eventually becomes hyper-polarized to prevent dangerous feedback loops from causing the neuron to continuously fire.

The spiking event occurs throughout the entire neuron, and is passed to neighboring neurons, again through synapses. Information in neural circuits is encoded in the timing and number of these binary spiking events. Most neural decoding aims to extract these spike occurrences and relate them to the activity of neural circuits, and therefore determine various neural states of interest.

**Neural Data Transforms**

BMI technologies are rapidly evolving to require higher channel counts to perform increasingly complex tasks with higher precision. These systems benefit from being fully implantable, however, fully implantable systems are constrained by wireless bandwidth and power budgets [55, 50]. Feature extraction aims to reduce the information bandwidth before data transmission or decoding by transforming neural electrical recordings into information-rich features. Long-term exposure to the neural environment on Multi-Electrode Array (MEA) implants can degrade the spike Signal to Noise Ratio (SNR) due to gliation and electrode degradation [53, 4, 47, 26]. This signal degradation undermines the accuracy and stability of many state-of-the-art feature extraction methods[18]. Implementing robust feature extraction methods that can operate reliably on degraded neural signals with low area and power cost is critical for the long-term viability of implanted BMI systems. The current state of implantable feature extraction from intracortical neural recordings can be categorized into three main methods: spike detection, spike sorting, and compression through broadband feature extraction [50].

The task of translating neural electrical recordings to understand the underlying neural states involves processing hundreds to thousands of channels of streamed neural data. To make such a task tractable, several methods have been developed to transform raw electrical recordings into information-rich features that are used directly in decoding as shown in Figure 2.2. Feature extraction focuses on transforming the 1 dimensional neural stream of each channel individually into some quantity which is representative of the neural behavior for this stream. This leaves only the task of decoding intended behavior from the neural activity at each channel's site.

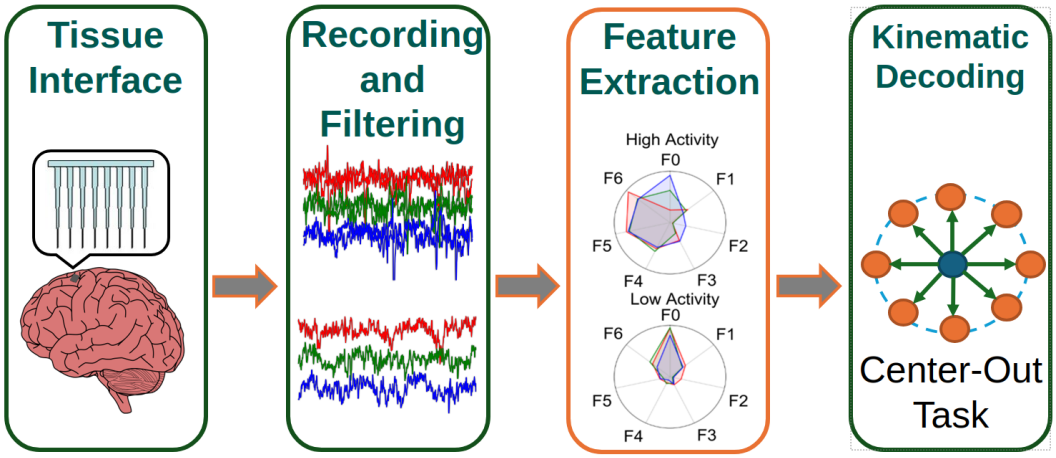



Figure 2.2: Pipeline for neural decoding.

**Spike Detection**

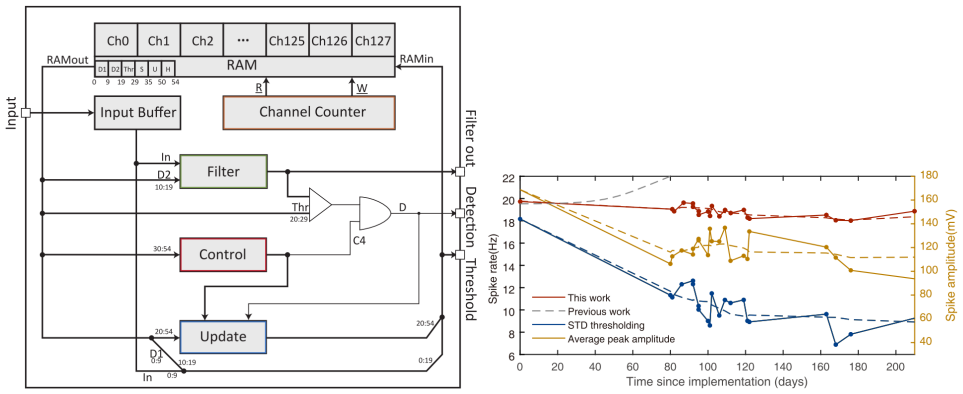
Spike detection is one such transform that aims to recognize individual spiking events often through identifying a threshold crossing of some quality of the neural electrical recording. Various qualities include raw amplitude, standard deviation from baseline noise, nonlinear energy operator, wavelet decomposition, ect. Real intracortical neural recordings are rife with noise and non-stationary dynamics. This makes spike detection in chronic implants without noise contamination difficult.

State-of-the-art spike detection has three main vectors of optimization. The first is robust detection of real spiking events while rejecting noise-induced false spiking events. The second is adapting to the non-stationary recording artifacts generally present in neural recording and digitization front ends. Lastly, is the power and area efficiency in hardware implementation techniques. The robustness of a neural decoding pipeline is significantly influenced by the spike detection algorithm used to identify the spiking event.

A naive approach to detecting neural spikes can be realized by simply setting a threshold on the voltage amplitude and recording every instance that the voltage crosses this threshold. This approach is incredibly unstable, since the noise power and gain can drift over time and between channels. Many spike detectors first use an emphasizer to magnify qualities in a spiking waveform out of the noise. Emphasizers often focus on the highly non-linear nature of spiking events, resulting in large spikes in high-frequency energy. The Non-linear Energy Operator and its variants (NEO) [15, 49, 67, 33], Amplitude Slope Operator (ASO) [67], and Energy of Derivative (ED) [33] emphasizers have been developed to magnify these characteristics, and are all relatively cheap to implement in hardware with their digital approximations shown in (2.1).

$$\begin{aligned} \psi NEO[n] &= x[n]^2 - x[n-1]x[n+1] \\ \psi ASO[n] &= x[n](x[n] - x[n-1]) \\ \psi ED[n] &= (x[n] - x[n-1])^2 \end{aligned} \tag{2.1}$$

Due to their relatively low complexity, these techniques have been used extensively in the spike detection step of neural data transforms with very low power and hardware requirements. Unfortunately, these emphasizers still suffer from the nonstationary nature of implantable neural probes. As probes degrade and move, the root mean square of noise as well as the SNR of the spike change. As a result, spike detection thresholds need to be adapted to levels that allow discrimination of the spiking event from noise. The architecture of [68] introduces a calibration-free and hardware-efficient method of spike detection. The architecture they derived is shown in 2.3 A, with the performance metrics in 2.3 B. This system uses an absolute difference filter as an emphasizer, and dynamically sets its threshold to target a heuristic firing rate determined by the expected neurophysiology of the implant. The system has a power and area cost of 6760 $\mu m^2$ and 0.038 $\mu W$ per channel, respectively, and showed maintained detection accuracy for 200 days. However, chronically implanted neural probes used in our study have mean noise levels of 89% six years after implant, which is much higher than the 20% tested in their work.




(A) (B)

Figure 2.3: System of [68]:
(A) System overview of calibration free spike detector. (B) System performance (red) compared to systems using other thresholding techniques (blue and yellow) and non-adaptive thresholding techniques (gray-dashed).

### Spike Sorting

While detecting a spike gives you information that a neural firing event has occurred, a single shank of a Utah multi-electrode array may receive neural spikes from 0-4 neurons [19]. These neurons may have different purposes, and so identifying which spiking event came from which neuron is of use to neuroscientists and neuro-engineers who want to decode the underlying neural state. The neural spike that we record on an electrical probe is a byproduct of the movement of ions in and around cell membranes to accommodate the spiking event. The spiking waveform received by a neural probe will therefore be affected heavily by the local neural environment. For this reason, a spiking waveform from one neuron may have a slightly different spiking waveform shape when observed by separate probes/locations in the neural medium. This phenomenon is used to isolate the spiking activity of individual neurons from each other, as well as separate real spiking events from noise. These techniques are colloquially known as spike sorting.

There are as many different spike sorting methods as there are calls in the forest, all with different nuances and tradeoffs, however they all primarily share a similar processing pipeline as depicted in Figure 2.4. Once a detected spike is identified through spike detection, a small bin of neural data is captured around the spiking event. Features of the waveform are extracted to project the neural event into a feature space. Spiking events are then grouped into different categories depending on their placement in this feature space. Some systems which experience high levels of inter-channel spike correlations, then cluster these groups based on correlations between the timing of different spike events. The additional complexity to the neural data transform process increases hardware requirements.

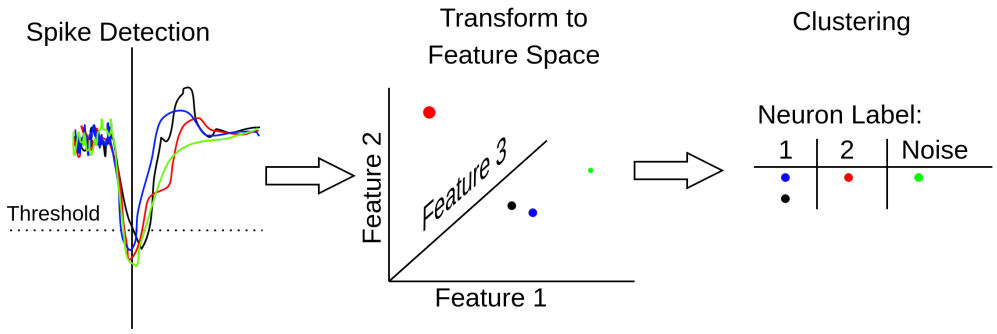



Figure 2.4: Simplified example pipeline for spike sorting.

Spike sorting can be very computationally taxing; however, several attempts have been made to implement spike sorting algorithms in hardware given at low area and power. The architecture in [12] explores the use of geometry-aware OSort algorithms to reduce spike sorting complexity, allowing the system to sort spikes of high-density neural probes with an average spike detection accuracy of 93.6%, and a clustering accuracy of 97.7%. This system only consumed 1.78 $\mu W$ of power and 1300 $\mu m^2$ of are per channel in 22nm FDSOI technology. Figure 2.5 depicts their system architecture with the three distinct steps of spike sorting: spike detection (CSD), spike feature extraction, followed by clustering. The major downside of spike sorting in the context of chronically implanted neural implants is in the reliance on spike detection in the first stage of neural processing, leading to poor performance several years after the neural probe is implanted [53, 47, 26, 4].

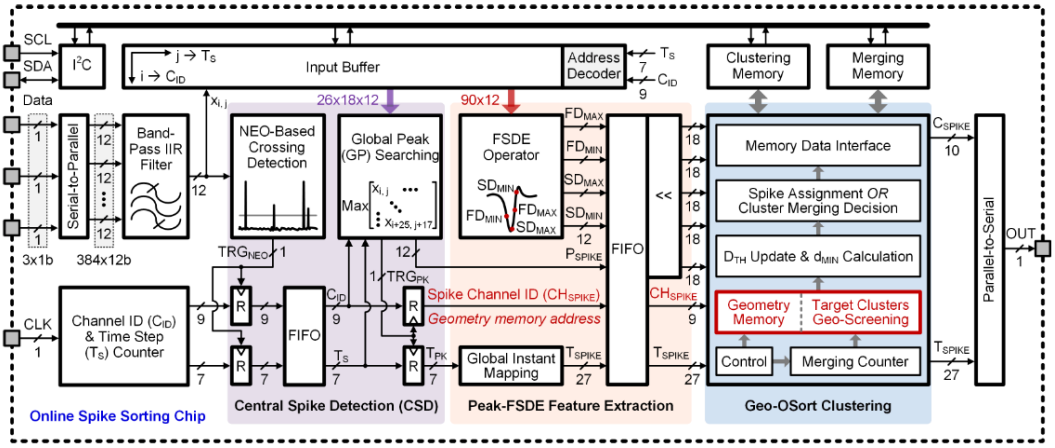



Figure 2.5: Spike sorting architecture developed in [12].

### Probe Degradation

The instability of neural implants is mainly driven by corrosion of implanted neural probes from exposure to the neural electrochemical environment [53, 4, 26], and the neural immune response which causes glial bodies to encase foreign objects. Multi-electrode arrays designed for intracortical neural recordings are subject to degradation over the long lifetime of an array [47]. Neural signal processing algorithms that use spike detection as a basis to their transform are likewise sensitive to the spike SNR, and even state-of-the-art spike detection and sorting algorithms show significant false positive rates at low spike SNR [7]. The SNR of our implanted arrays is quantified in (2.2)

$$SNR_{spike} = \frac{|max(Spike\ Waveform)|}{\sigma_{baseline}} \tag{2.2}$$

where the spike waveform is the wavelet identified by spike detection and $\sigma_{baseline}$ is the standard deviation of the baseline non-spiking data.

Using the same spike-sorting analysis as [64], spikes were detected and sorted based on 2-4 PCA components of the spike waveform. Based on these extractions, the maximum SNR of the spikes were determined using equation (2.2) over 50 open-loop sessions from 2019-2022. The count of spiking units with SNR greater than 4 (the threshold for spiking units that are considered good enough for spike sorting [64]) over time is shown in Figure 2.6. It is evident that the quality of neural recordings degrades rapidly years after the multi-electrode array is implanted. Such noisy neural recordings prove difficult for stable and accurate kinematic decoding with conventional feature extraction methodologies which rely on spike detection and sorting.

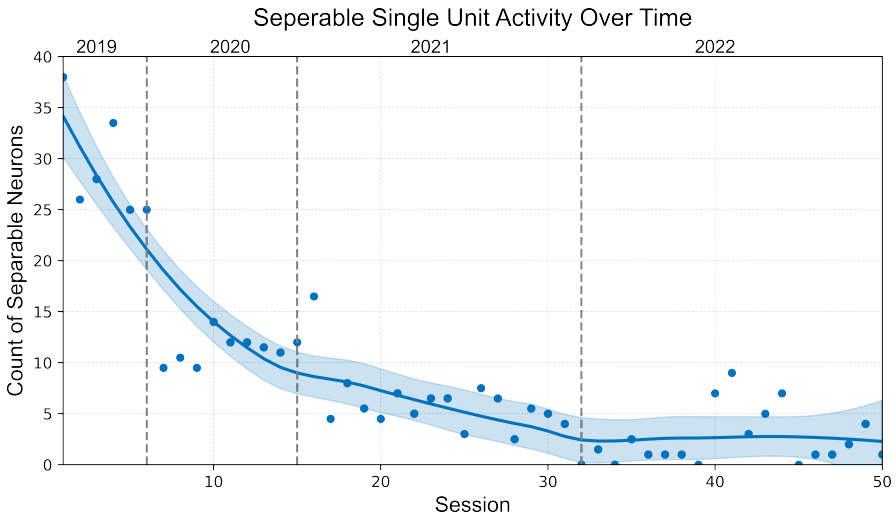



Figure 2.6: Count of neural spiking units with SNR greater than 4. LOESS fit shows the average trend spiking unit quality.

### Broadband Feature Extraction

As the neural environment degrades the interface between electronics and the neural circuits, spike detection methods belie their frailty. The inherent non-linear nature of transforming continuous electrical activity to event-based encoding causes neural spiking events that are long lost to noise to be discarded entirely. Therefore, chronically implanted systems benefit from a more linear approach; either simply compressing the neural waveform for complete analysis by external computation hardware, or linearly relating features to a 'total neural activity' which aggregates the presence of neural features without threshold discrimination. Broadband feature extraction focuses on efficient summarization of ensembles of neural activity rather than prioritizing the discrimination of activity between neural units. This has been found to be significantly more robust to probe degradation and therefore more relevant for long-term BMI applications [23, 24].

One of the simplest examples of broadband feature extraction is to summarize the band power most associated with neural spikes, referred to as spiking band power [3, 54, 40]. Spiking band power has remarkable simplicity as it only requires taking the mean absolute value of the neural signal to recover a neural feature. The work in [40] showed that this feature extraction method can aggregate the spiking activity of very low-amplitude spike waveforms buried within the noise. An example of spiking band power feature extraction is shown in 2.7. Without applying thresholding, the spiking band power is aggregated to create a feature which is correlated with spike firing rate, but does not miss noise dominated spiking activity. This feature, however is still sensitive to noise, as it does little to reject non-spiking activity from corrupting the feature space.

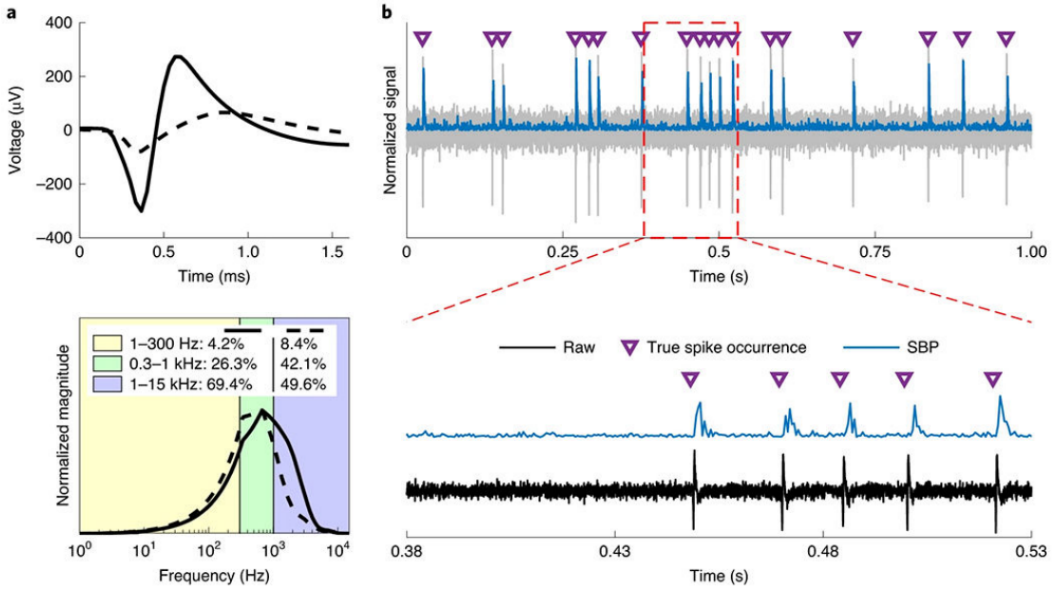



Figure 2.7: Representation of spikes within the 300-1000 Hz band from [40]. (A) Frequency spectrum of two averaged neural spikes. (B) Simulated neural activity with spiking band power overlaid in blue.

## 2.2 FENet Algorithm

To address the difficulties of extracting neural features in the highly noisy environment of chronically implanted neural interfaces, FENet [22] adopts a seven-layer architecture inspired by the Daubechies-20 (db20) wavelet transform, with additional non-linearity and accumulation components. Accumulating the wavelet power of data-driven kernels enables the summarization of low-amplitude spiking activity, while rejecting the noise inherent to spiking band power features. Tailoring this transform to neural data improves the performance of feature extraction and allows end-to-end training of the neural decoding pipeline. The resulting features have demonstrated state-of-the-art performance in high-noise recordings from chronically implanted multi-electrode arrays. We show the full data processing pipeline in Figure 2.8.

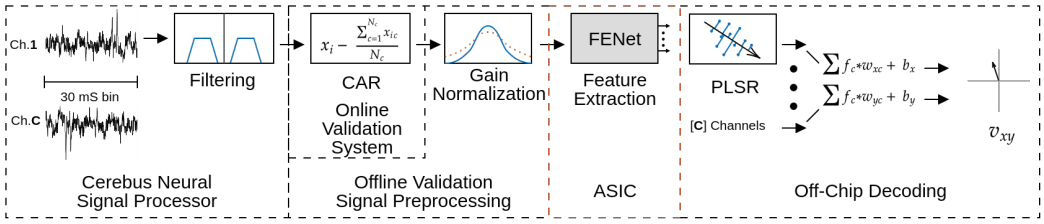



Figure 2.8: Feature extraction and decoding pipeline used with FENet models.

This pipeline describes the digital signal processing steps used to transform digital recordings of Utah arrays into decoded kinematic intent. The electrical recordings are first filtered using bandpass filters with cutoff frequencies of 0.3-5 kHz. To remove common-mode noise, the recordings are common-average-referenced by subtracting a common average across probes from each timestep. The mismatch in signal gain caused by different digitization hardware for each site can be removed by gain normalization. FENet feature extraction is then used on these processed signals to concentrate important information in the neural recordings. Using these concentrated representations of neural activity, the neural state is finally decoded. This dissertation focuses on the implementation of the FENet feature extraction step in hardware and is the only step in the decoding pipeline implemented in ASIC hardware.

The data conditioning steps preceding feature extraction follow standard practices in BMIs and are already demonstrated in existing hardware systems [35, 3, 32]. Partial Least Squares Regression (PLSR) is used to reduce the number of output feature dimensions, preventing overfitting of the decoder. PLSR is trained per channel on data from a single day, then model parameters are averaged across all channels to generate a single transform used for all subsequent days, and is general to all channels.

The FENet algorithm is shown in Figure 2.9. The construction of the algorithm is different from most convolutional neural networks. Its topology is similar to a discrete wavelet transform with the addition of leaky ReLU nonlinearities, pooling, and data-driven kernels. Each layer has two sets of outputs with the first set of outputs passed to the next layer without any additional non-linearity in between, and the second set of outputs pooled within each layer to generate features.

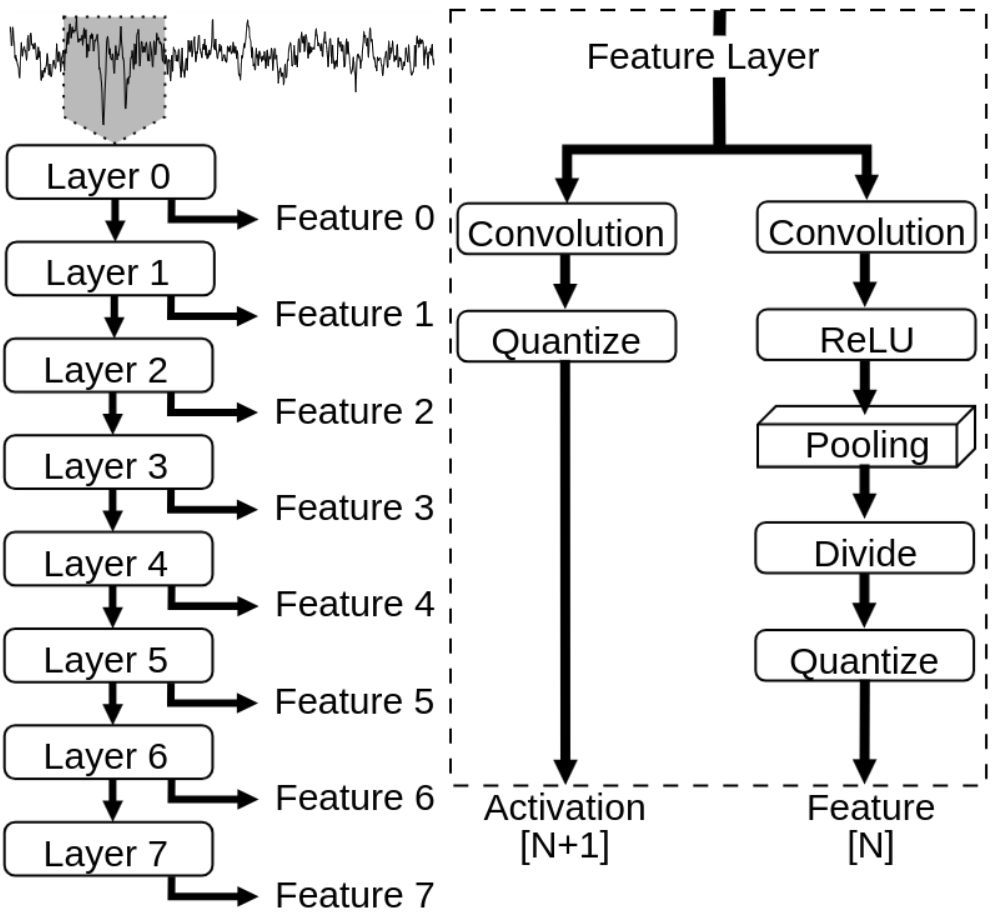



Figure 2.9: FENet algorithmic flow: (Left) Multi-layer data flow for FENet on a single neural channel. (Right) Internal computation within each layer.

Each layer performs two separate 1D convolutions on its input stream with the traversal and feature-generating kernels. The traversal path (left) generates an intermediate output passed to the next layer. The feature-generating path (right) applies a Leaky Rectified Linear Unit (LReLU) non-linearity followed by global average pooling through accumulation and subsequent normalization through division. This architecture allows summarization of kernel power at different levels of resolution, providing information about highly local wave shapes at the lower levels with diminishing degrees of locality as the number of layers increase. The output feature computation is defined in (2.3):

$$f_l = \frac{\sum_{i=0}^{\lceil \frac{B}{S_l} \rceil} LReLU\left[ \sum_{j=0}^{K_l} x_{S_l * i - j} \cdot w f_j \right]}{D} \tag{2.3}$$

where $f_l$ is the output feature from the $l^{\text{th}}$ layer. Each layer receives a bin of $B$ samples, which is convolved with kernel weights $w f_j$ of width $K_l$ and stride $S_l$. The LReLU output is accumulated and quasi-normalized using a division factor $D$. The leak factor of the leaky ReLU nonlinearity is a trainable hyperparameter with values constrained to the set of negative integer powers of two as defined in (2.4)

$$\frac{-1}{2^{-n}} \ \{n \in \mathbb{Z} \mid 0 \leq x \leq 22\} \tag{2.4}$$

. The traversal path computation, defined in (2.5):

$$X_{L+1} = \sum_{j=0}^{K_l} x_{S_l * i - j} \cdot w t_j \tag{2.5}$$

produces the intermediate activation $X_{L+1}$ for the next layer, using traversal weights $wt_j$. Kernel sizes are matched across both computation paths to reduce complexity. The overall structure of the hardware-implemented algorithm has a single continuous stack of successive convolutions, lateral convolutions in each layer which extract features at successively larger resolution scales.

This algorithm requires constant processing of neural streams, with many thousands of multiplications per feature. While the success of FENet makes this algorithm alluring, implementing it within the low power and area constraints of implantable devices is a significant challenge. This thesis explores the requirements of the algorithm, reduces its complexity, and implements it in 65 nm CMOS with a conscience towards minimizing power and area. While this is the first implementation of 1D CNN feature extraction for neural interfaces, other general purpose architectures have been developed for their own purposes. In the next section, the space of CNN accelerators will be explored to understand what gaps exist in this space which need to be filled to efficiently implement the FENet workload.

## 2.3 CNN Accelerators

Convolutional neural networks have shown prowess in identification and categorization. Their viability in these assignments has spurred diverse development in architectures designed to efficiently and performantly compute these algorithms. CNN ASIC implementations must balance a plethora of merits to best fit their application. Show in Figure 2.10 is a representation of these merits reminiscent of the design octagon of analog amplifiers introduced by Behzad Razavi [44]. This diagram highlights the design priorities for the FENet workload within the context of brain machine interfaces, and provides motivation for many of the design choices discussed within this dissertation.

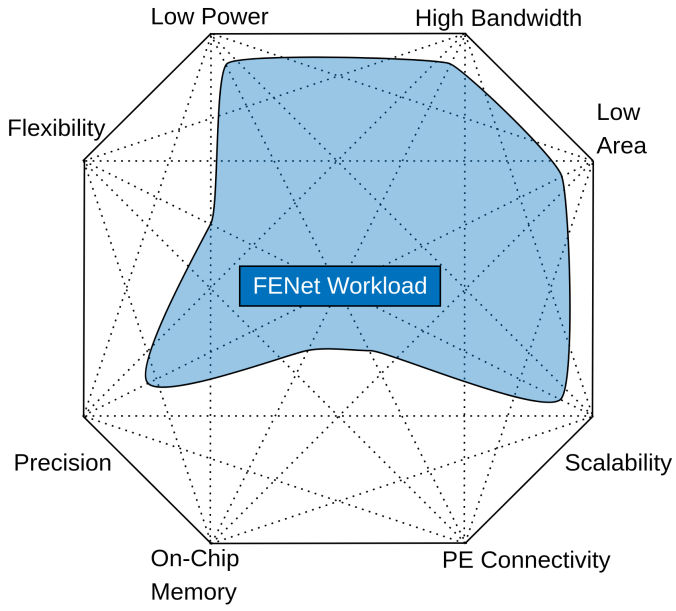



Figure 2.10: Design octagon for deep neural network processing architectures. Qualitative merits for FENet workload indicated with blue shaded area.

Like in the case of amplifiers, the 'best' architecture for a particular system is the one that maximizes fitness to the needs of the intended workload. Processing neural data has a fitness profile that is significantly different than the multitude of CNN ASIC processors in the current literature. These differences stem from both the physical nature of the recording systems and the statistical properties of the signals themselves.

Designing a CNN processor for neural data requires careful attention to the unique demands of this domain. First, neural signal processors must support extreme scalability to accommodate the ever-growing number of recording channels in modern neural interfaces. Since these systems are intended for implantable devices, where thermal and energy budgets are tightly constrained, low absolute power budgets are a necessity. Further, to support integration near the data sources, the processor must also maintain a small silicon footprint, minimizing both die area and cost for commercial viability.

Specific to the FENet workload, each feature set has very little activation reuse between channels. This highlights the importance of one-to-one channel parallelism, where each channel's data is processed independently and in real time as it arrives. Processing data in time as opposed to in batches, reduces or eliminates the need for large memory caches and enables computation of streamed data while minimizing the on-chip memory footprint.

Unlike general-purpose CNN accelerators that aim for broad model support, the design here requires algorithmic flexibility within the FENet archetype, allowing for hardware specialization without full generalization to other CNN models. Similarly, numeric precision must be finely tuned: low enough to save power, but high enough to preserve kinematic decoding accuracy, striking a delicate balance between efficiency and fidelity. In terms of throughput, while the absolute number of operations per second may be lower than that of image-oriented CNN applications, the opportunity for activation reuse between channels is zero and minimal within channels, meaning computation cannot benefit from activation reuse as is typical for 2D CNNs. Finally, biological neural signals are highly non-stationary. A large proportion of channels often contain only noise or irrelevant features at any given time. For this reason, a neural CNN processor must proportionally scale its power with the number of informative channels it processes, ideally reducing total power by depowering uninformative channels. In this work, the processing element, cache, and data network connectivity are optimized to efficiently implement FENet.

### Data Networks

An architecture's network design is one of the most important factors in determining the utilization of processing elements and memory access power. On one end of the design space, highly interconnected processing elements and memory channels allow for significant reuse of activations at the expense of overhead and increased complexity, which leads to scaling difficulty. On the other end, each processing element has its own dedicated memory channel, which constitutes a high-bandwidth, low connectivity architecture. For image processing, the former is ideal since each activation can have hundreds of reuse opportunities. FENet, on the other hand, has orders of magnitude lower reuse opportunities for each activation, with high batch parallelism. For this workload, high bandwidth is a priority to allow 100% utilization of each processing channel.

Figure 2.11 depicts the various types of networks for deep neural network processors. The network topology of a deep neural network processor defines two key aspects of an architecture: the bandwidth, and the processing element inter-connectivity. If the hardware resources for networking are kept constant, then these merits become antagonistic where increasing the performance in one aspect, reduces the performance in the other. Figure 2.11 shows the spectrum of network connectivity that a deep neural network processor can utilize with high bandwidth, low connectivity topologies on the left, and high connectivity, low bandwidth topologies on the right. For the FENet dataflow, the parallel processing of many independent channels indicate that the same operation will be repeated at large scales with little opportunity for reuse. In the following section, the workload for different deep learning models will be presented with discussion on the benefits each network topology brings to various forms of dataflow. For reasons that will be elucidated in detail within this section, the network topology of this ASIC was chosen to be a unicast.

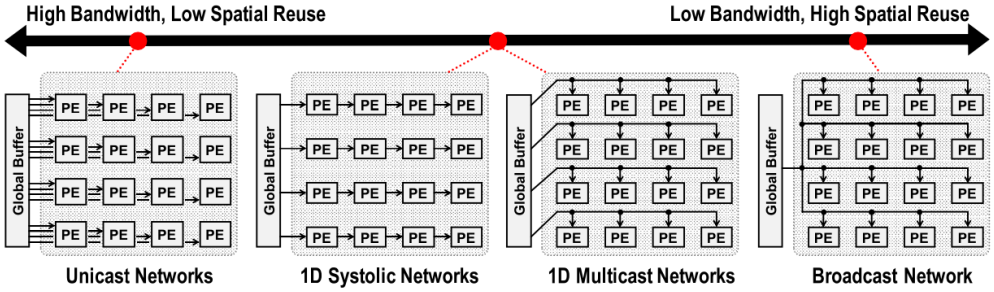



Figure 2.11: Common network types for CNN processors [13].

**Dataflow Types**

A dataflow type in the context of deep neural network processing is an architectural characteristic that defines how data flows through the processor. Since the objective of low power deep processing is to reduce the amount of data movement as much as possible, these characteristics are named after which computational component is held locally near processing resources. The taxonomy of deep neural network processing topologies are shown in Figure 2.12. These topologies represent the general way in which operations can be mapped onto data and include input stationary, weight stationary, output stationary. A given workload will have an optimal dataflow type which implements it, and each dataflow type is better suited with certain network topologies and bandwidth requirements. If the system is intended for a single workload requirement, designing the architecture to best accommodate its dependency graph will primarily determine the system's efficiency. Hybrid taxonomies such as row stationary look to combine the benefits of each topology at the cost of hardware complexity. The following section will provide detail to which topologies are appropriate for a given workload and which data networks are optimal for each topology.

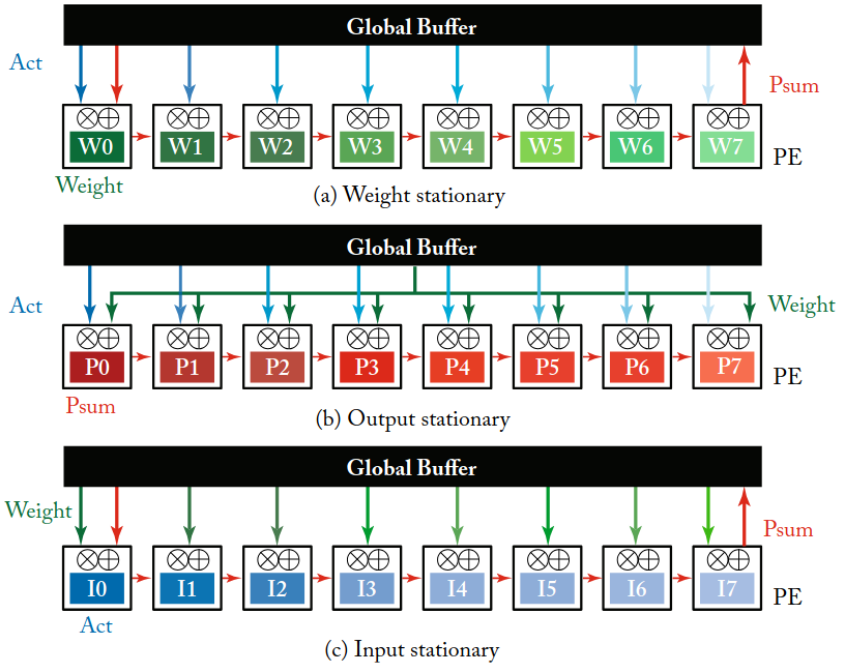



Figure 2.12: Taxonomy of common deep neural network accelerators. *Act* means activation and different shades of the same color are used to represent different values of the same type of data.

Input stationary architectures like [42] keep inputs local to processing elements as weight filters are streamed in and partial sums streamed out and generally employ a high-bandwidth (unicast) weight distribution networks with systolic transport of partial sums. This dataflow type is optimal for workloads with a high number of filters compared to input channels. Architectures with weight stationary flows such as [61] store weight filters locally, and are optimal for workloads which use the same filters over a high number of input channels. Weight stationary flows benefit most from systolic networks as partial sums from one processing element are pipelined immediately to its neighbor. Output stationary architectures [17, 52] store partial sums locally, benefiting workloads which have a high number of independent channels in which partial sum caching would become onerous. As such, output stationary flows pair well with unicast data networks since each processing element can access its own channel of memory.

Hybrid dataflow approaches try to minimize data movement along multiple tensor dimensions. The well-known Eyeriss V1 [14] architecture adapts a row stationary dataflow which locally caches the row of both inputs and weights. This architecture generally employs 1D multicast networks for broadcasting rows of both weight and activations. Although highly efficient for large 2D CNN workloads like AlexNet, these additional complexities greatly increase resource requirements, and sacrifice processing element utilization for highly batch parallel workloads.

Many of these architectures struggle to efficiently map a workload like FENet onto their processing resources because of its minimal activation reuse [13] and the high batch parallelism necessary to neural decoding. The solution is to customize both the dataflow and network to best serve the FENet workload while minimizing the excess hardware and power consumption to make it suitable for implantable brain-machine interfaces. A unicast input activation network is selected to maximize the utilization of processing elements since each channel is independent from its neighbors. Weights are broadcast globally since they are shared among all channels. A novel dataflow is constructed to hold each layer's feature stationary to the processing element, while generating intermediate activations and writing them to a nearby SRAM channel, reducing data movement.

*Chapter 3*

# SYSTEM ARCHITECTURE AND DESIGN

This chapter delves into the specific architectural design process of the FENet ASIC. Exploration of this architecture begins with a system overview and then zeros in on each major component of the design. The design of this architecture balances the three design goals of low area and power without compromising decoding performance.

## 3.1 Overview

The proposed 1D-CNN stream-oriented processor is designed to extract features from high-bandwidth neural data streams in a scalable and configurable manner. The system is a broadband feature extractor which outputs 2-8 values per channel that represent the aggregate presence of learned feature kernels in the neural data. Like other broadband feature extractions, these features are able to represent the presence of low-amplitude neural activity from highly noisy signals that would otherwise be lost to spike detection and spike sorting feature extractors. The architecture is optimized for stream processing with minimal activation caching since it schedules operations based on the availability of input data and completes each convolution in a piecewise manner between slowly arriving neural samples. This allows scaling from very few, to hundreds of neural channels to be processed in real-time with minimal control and memory overhead.

The ASIC hardware components can be classified into two main groups: the CNN solver hardware under test, and the validation system. The former consists of the channel block macro, algorithm control finite state machine (FSM), processor control FSM, and configuration registers. The latter is the custom serial data interface, which exchanges data with the external validation system.

The system diagram is shown in Figure 3.1, which illustrates the data flow of the system and channel architectures. The blocks on the left denote the centralized control hardware which generate control signals that are broadcast the the processing hardware. At the top left is the data interface hardware which accepts data and configurations and imports them into the computational system. Below this block are the configuration registers that store configurations and influence the control sequences generated by the algorithm and processing element (PE) control finite state machines, which generate control signals to modify the behavior of the processing elements to match the desired model configuration. Control signals and data are broadcast globally to all processing hardware and are color coded based on function. The interface data bus in red writes input data to the asynchronous queue of each enabled channel. Configurations like channel and layer enable are distributed to each channel from the configuration registers (colored blue).

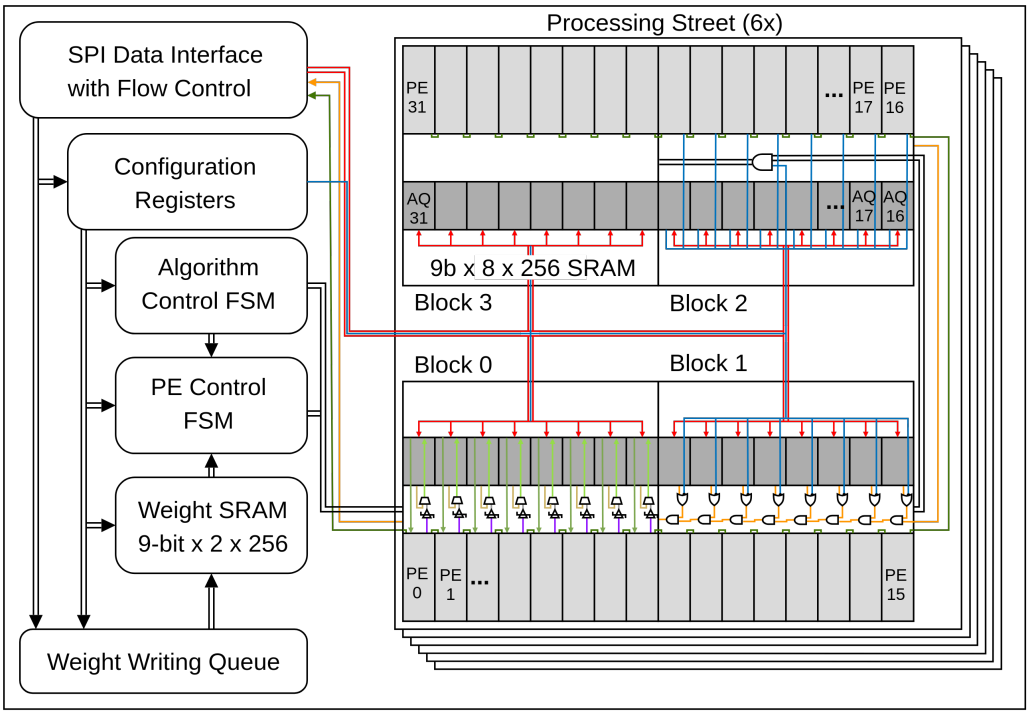



Figure 3.1: Functional representation of ASIC system architecture showing their logical connections. Various buses are color coded: Interface data bus (red), algorithm and mac control (black), data available (orange), feature out (green), and channel enable (blue).

Controls, denoted with double black lines, are generated from the two finite state machines and broadcast to each processing street. Each block depicts a different aspect of signal distribution. Block 0 shows how data arrives at the write port of the SRAM circuit. Outputs generated by the processing elements (PE), and level shifted to the memory voltage domain. The data source is selected between the asynchronous queue (AQ) for writing neural data and the output of the processing elements when writing intermediate outputs. Block 1 depicts the generation of a global data available signal in orange from the data available flag generated at each channel. The inverse of the channel enable signal is or'ed with the data available signal of each channel. This disables its dependency when the channel is not enabled. A globally reduced buffer full signal is generated in a similar way to indicate when data loading must be stalled. Block 2 indicates how control signals are buffered from one block to another using skew-optimized AND gates. If the blocks above these buffers are not enabled, the buffers are disabled to reduce switching power. Features generated by processing elements are exported via a single bit scan chain in green. Disabled channels are skipped in this scan chain with a multiplexer.

Processors that share a common SRAM memory are grouped into channel blocks, and channel blocks that share a buffer chain and activation bus are further organized into streets. The data interface serves as the primary access point for configuring registers, loading weight memory, and streaming neural activations into the system. Multiple clock and voltage domains, as well as channel and layer level power gating are used to minimize power.

We enable a high degree of parallelism without excessive area overhead by utilizing word-serial processing between each system clock cycle. The use of separate clock networks allows the high speed processing element clock (MAC clk in Figure 3.6) to be constrained to a lower VDD domain, reducing switching power. The processing element (PE) control FSM synchronizes control signals to the rising edge of each system cycle, and does not begin a new control sequence until the next rising edge of the system clock, preventing control misalignment. The frequency ratio is adjusted for model parameters that require more MAC cycles.

The system utilizes a third asynchronous interface clock for IO to emulate data from independently timed sources. Neural data is distributed to each channel via the data interface by a shared bus. Although the data interface is utilized to compensate for the limited off-chip IO of the validation ASIC, the interface data bus is intended to be replaced by parallel data sources within a full chip BMI decoding system. For this reason, each channel is equipped with a 4 element Asynchronous Queue (AQ in Figure 3.1), to allow for neural data caching while the SRAM is in use for computation. These asynchronous queues indicate their fullness state with *data-available* flags (orange) which are reductively ANDed together to indicate the system's readiness for processing.

The presence of data in all enabled channels signals the central FSM to trigger the load operation of the first layer. This operation transfers neural data from the asynchronous queue into the first layer's SRAM space. Once a stride of data is loaded within a layer's memory space, higher-order dependencies are satisfied, and the processing element becomes available, data within a layer's SRAM space is read back and presented to the processing element for computation.

At the completion of feature generation, the pooling register value is reduced to 9 bits, rounded, and added to the feature shifting register (Feature SR). At this point, the processor begins a new computation, while the feature is shifted out of the processing element. Each channel is equipped with an always-on skip multiplexer that allows its position in the feature scan chain to be skipped in the event the channel is powered down. The features are shifted out, and returned to the data interface for exporting off the chip.

The proposed architecture is designed to efficiently map a wide range of FENet models. Kernel size, stride, LReLU leak slope, and pooling parameters are all configurable through the CNN control FSM sequence. Up to 8 features can be generated from 7 feature-generating layers and one terminal traversal layer. Kernel size is limited only by the depth of the weight SRAM (256 elements) such that the sum total of traversal kernel weights must be less than 256 (the total number of kernel weights is 2$\times$ traversal weights). Bin sizes are defined by multiplying the first layer stride with a programmable cycle counter, allowing maximum bin lengths up to 2048 strides. These options provide a wide hyperparameter space for optimizing power, performance, and decoding stability.

To support system scalability, the processing hardware and activation/feature caching are integrated into a modular channel block macro. Each channel block contains 8 processing elements , each with their own asynchronous data queue, gated control-signal buffers, and power-domain level shifters. All processing elements share a customized TSMC 72-bit $\times$ 256-element low-leakage single-port SRAM macro. To accommodate the higher voltage headroom requirements of the proprietary SRAM, we separate the memory and processing element VDD domains. Shown in Block 0, outputs from each processing element are level shifted from the processing element domain (MAC VDD) to the SRAM (MEM VDD) domain. Single-port SRAM macros are used for compactness and power efficiency, with write access multiplexed between asynchronous activation queues and processing elements.

## 3.2 Processing Element Architecture

The processing element is the heart of any CNN processor. The following section will detail the plethora of design considerations that went into tailoring this processing element to the FENet dataflow, while reusing every component as much as possible and maintaining full configurability of the FENet algorithm. Control signals and weight data are broadcast globally by a centralized FSM, orchestrating synchronized computation across all processing elements.

### Channel Architecture Overview

To provide context for processing element design, this section starts with the overall architecture, and a brief synopses for what the intention is for each sub-component. The following subsections elaborate on the reasoning behind each choice. As shown in Figure 3.2, each channel contains two fused MAC data paths, the feature path, which writes partial sums to one of the seven pooling blocks, and the traversal path, which handles intermediate feature computation between layers. For lower-order layers, traversal partial sums are written back to SRAM to serve as inputs for higher-order layers. For the highest-order layer, the traversal output is instead accumulated into the final pooling block, producing the last feature output. Data read from the SRAM is operated on in the data path blocks. The outputs from the data path are shifted and added into the pooling layer shift registers depending on the active layer. The feature shift register within each layer latches the output feature for shifting out of the processing element while a new computation commences.

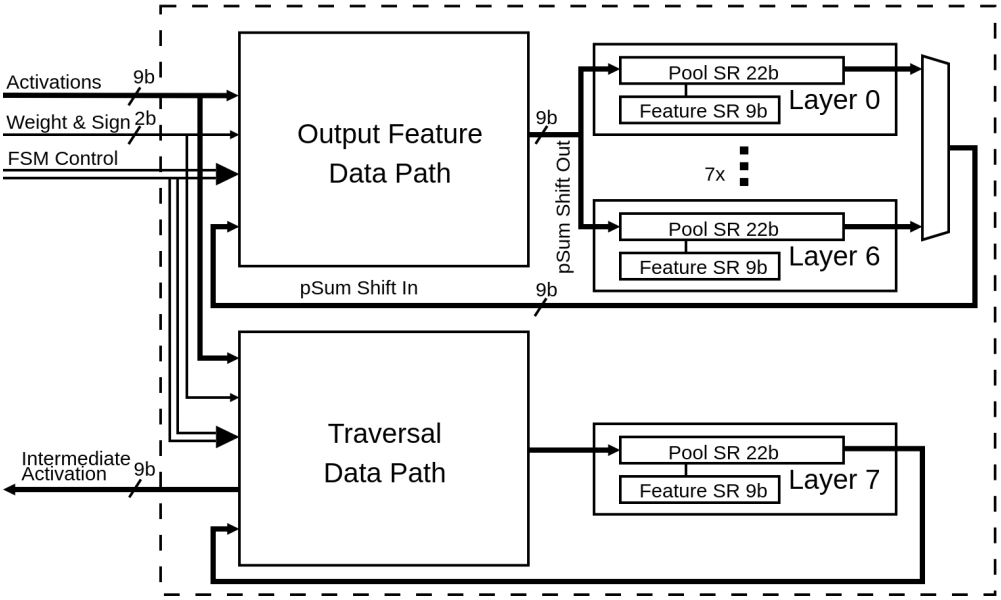



Figure 3.2: Architecture for one neural channel. Two arithmetic units simultaneously process the traversal and feature generating data paths. Intermediate values are passed to pooling accumulation register blocks, selected by a multiplexer.

Each processing element is designed to meet strict area and power constraints critical for large-scale neural decoding systems. To minimize the hardware footprint, multiplication is performed using a word-serial approach, allowing an 8-bit adder to be reused across all CNN operations. This design choice significantly reduces the area cost and enables scalable integration of hundreds of channels, at the cost of some increase in switching power relative to parallel logic architectures.

To ensure robustness during long accumulation sequences, overflow handling is incorporated. An auxiliary overflow bit allows temporary excursions beyond the nominal accumulator range without immediate clamping. Final clamping is applied only if the terminal convolution result remains out of bounds.

The processing element clock frequency is configured relative to the system clock to guarantee that all multiply-accumulate, nonlinearity, and global average pooling operations finish within each system clock interval. This clock ratio is tunable across models to optimize performance and power efficiency.

**Data Representation**

The first step in scoping the design of an arithmetic system is to define the representation and resolution constraints of the data path. For this system, a fixed point format was determined to be ideal as it allows systematic discarding of least significant bits, maintaining a low bit-width requirement on the data path, without the overhead of floating point arithmetic. To determine the optimal resolution for the system, the FENet algorithm was quantized using QTorch [66], then simulated and validated for $R^2$ decoder performance. The results of this experiment are shown in Figure 3.3.

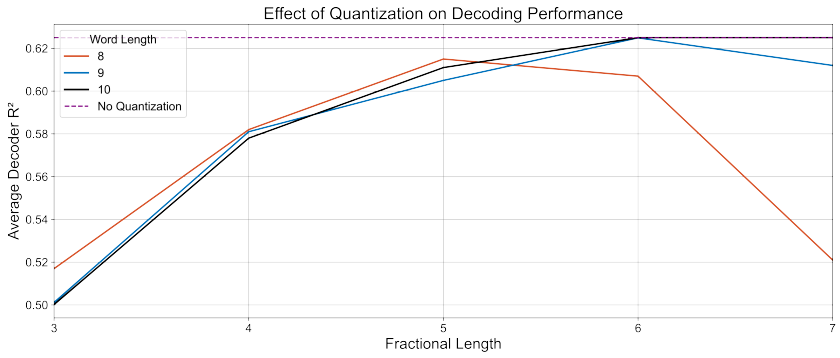



Figure 3.3: Effect of fixed point quantization of weights, inputs, and intermediate activations on the decoding performance of neural data. Dotted line in purple is the reference performance with no quantization applied.

The optimal values for the integer and fractional components of the system were determined to be 3 integer bits integer, and 6 fraction bits as depicted in Figure 3.4. Lastly, sign magnitude representation was used to reduce the switching activity of read and write operations in SRAM [11].

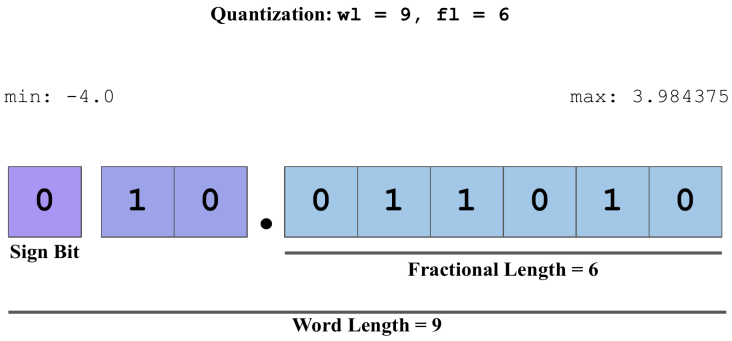



Figure 3.4: Fixed point representation chosen for the activation and weights of the FENet dataflow.

**MAC Architecture**

When choosing the architecture of an arithmetic unit, both power, and area are important considerations. While separate multiply and accumulation hardware allows word-level processing of multiplications in a single cycle, it comes at a significant cost of hardware. These architectures are depicted in Figure 3.5. The tradeoff between area and power is listed below. While the power performance of the MAC unit is decreased by 37%, the area tradeoff allows for improved area efficiency. Furthermore, the serialized MAC can incorporate bit-wise gating for zero-value weight bits. Analysis of the FENet weights showed many near-zero weights making the word-serial MAC architecture ideal for the system. Ultimately, the fused word-serial architecture was chosen since it also allowed for integration of word-serial accumulations of the pooling registers, reducing routing requirements and overhead. This decision, however, does come at the expense of an increased power consumption of the MAC.

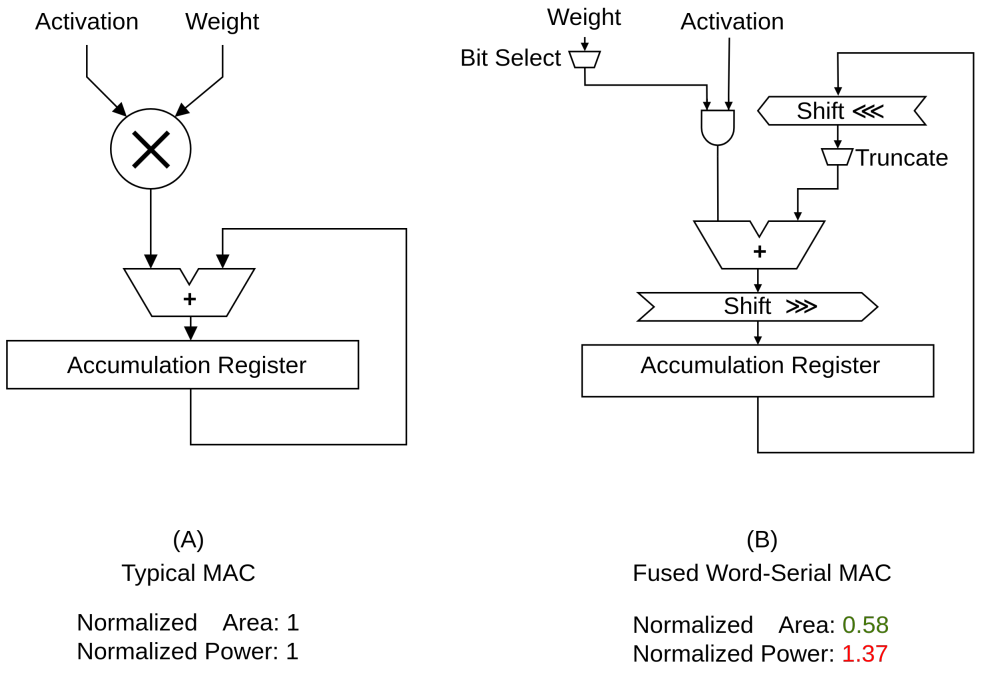



Figure 3.5: Comparison of two MAC architectures with normalized area and power tradeoffs. (A) Typical single cycle MAC unit. (B) Word-Serial MAC Unit.

A block diagram of the fully integrated MAC architecture is shown in Figure 3.6. At the center of the design is a single 8-bit adder shown in purple. This one block handles all arithmetic operations in the processing element, making it the centroid of the design. Starting on the left side of the datapath, the first operand to the adder comes either from the SRAM read port or from the shift-out port of the pooling block. The turquoise logic around it handles operand negation for subtraction during signed multiplication and sign adjustment during the leaky ReLU operation. On the right side, we have the shift-and-insert circuitry in green, paired with the output shifter of the accumulation register in yellow. Together, these decide which part of the accumulator feeds into the adder as the second operand. Looking at the output, the accumulation sum can either be added to one of the 7 pooling blocks, or written back to SRAM for higher order layers. The 8-bit adder is reused across multiplication, leaky ReLU application, rounding, overflow handling, and pooling accumulation. Pooling division is efficiently implemented through bit-shift operations.

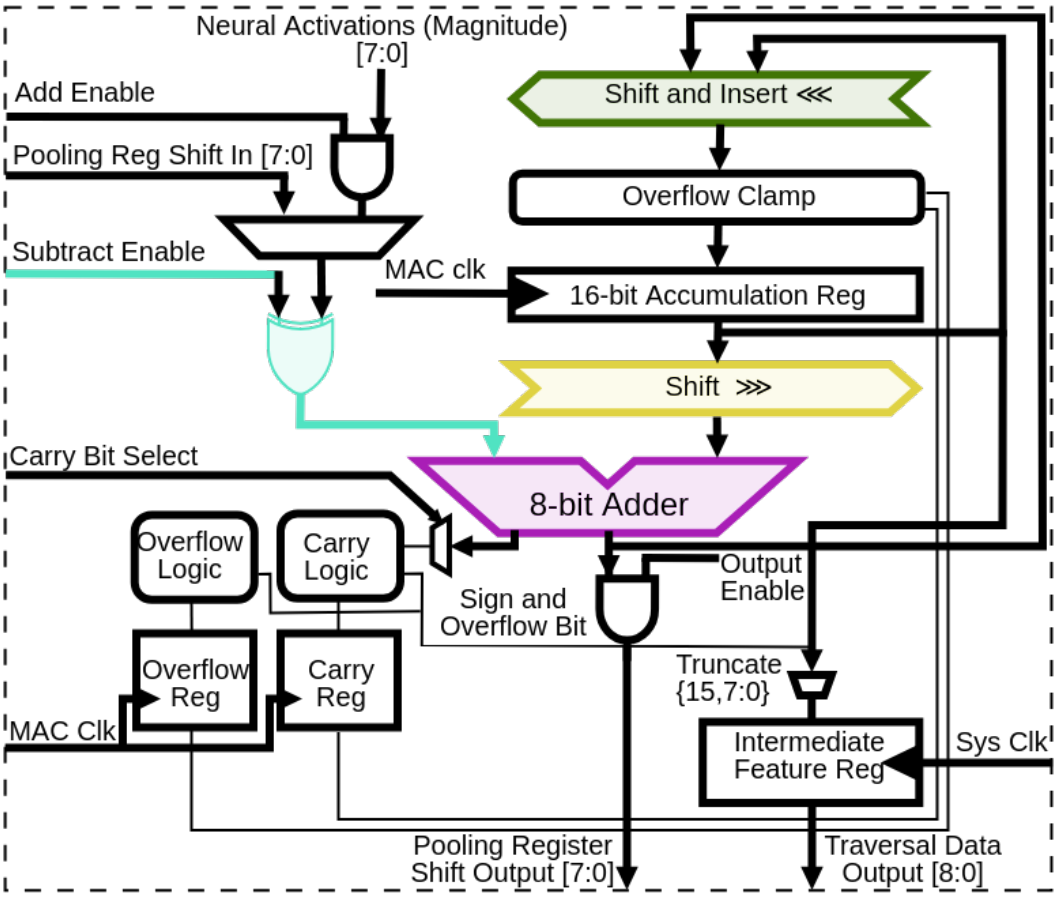



Figure 3.6: Architecture of single data path processor.

For more context on how this works, the flow of data is shown in Figure 3.7, with colors corresponding to hardware in Figure 3.6. In this computational phase, the 4th weight bit is currently selected, which shifts the active range of the accumulator by 3 bits. These 8 bits of the activation are added to the current accumulator, starting at the 4th bit place. The carry out bit is saved and added to the next phase, where the active window is shifted by 8 bits to complete the 16 bit addition. Partial sums are latched in a shifted window of the accumulator, and carry propagation is handled in subsequent phases.

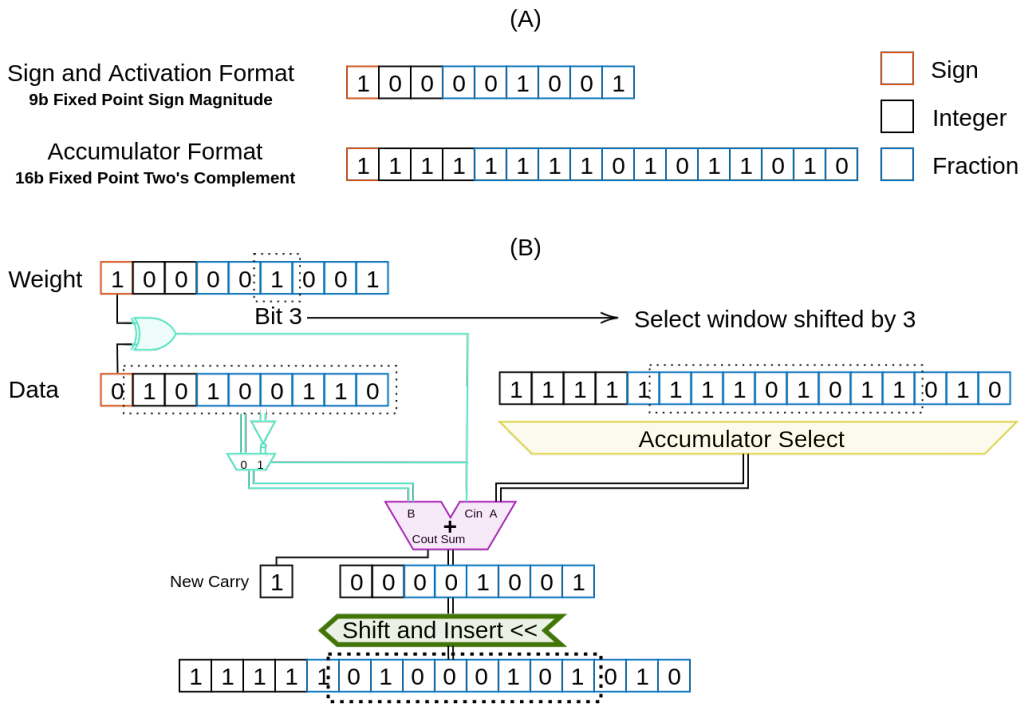



Figure 3.7: Processing element computation flow: (A) Data format and bit type key for 9b sign-magnitude activation and weight data as well as the 16b two's complement accumulator format. (B) Depiction of the addition of a single partial sum to the accumulation register at the start of multiplication phase 4.

The data path depicted in 3.7 shows how weights and activations flow through the system. Two weight bits (one for the output and another for the intermediate feature generating paths), are read from the centralized weight SRAM, are broadcast to all processing elements. At the same time, shift control signals are similarly broadcast to instruct the accumulator select, and shift and insert multiplexers to select the current active region of bits which correspond to the bit position of the weight bit. The adder is then enabled if and only if the corresponding weight bit is 1. This adds the activation to the accumulator within its current shifted position. The adder then finishes its addition, to carry the remaining bits. When the weight and activation are of opposite sign, the activation is inverted, and a bit is added into the carry-in to enable subtraction of the activation value.

At the completion of a convolution kernel, the accumulation register is reduced to bits 6 through 13. Rounding is completed by adding bit 5 to the remaining 9 bits, which helps alleviate some error from quantization. For the traversal path, the accumulation value is written to the `Intermediate Feature Register`. The value within the `Intermediate Feature Register` is written back to SRAM on the next system clock cycle. For the output feature generating path, and the traversal path when the last layer is active, the accumulation value is instead accumulated in an associated pooling block, which is selected based on the CNN Control signals. During pooling, the activation is instead multiplexed with the 8 least significant bits of the currently active pooling register. The output of the adder arithmetic is then redirected into the shift port of the pooling register, such that the sum is shifted into most significant bits of the pooling register.

### Overflow Clamping

A well-designed processing element must gracefully mitigate the effect of its limited dynamic range. A naive approach is to incorporate as long of a bit width as necessary to never overflow your arithmetic operations. This approach quickly balloons the hardware requirements of your system, without significant benefit from the increased resolution. The overflow clamping has two stages as depicted in Figure 3.8. The first stage is a soft-overflow, where guard bits are used to allow for momentary overflow into an unrepresentable range. During soft overflow, three things can happen, either the accumulator value returns within an representable range (Figure 3.8 A), the accumulator does not overflow, but still remains outside of the representable range of values (Figure 3.8 B), or the accumulator completely overflows (Figure 3.8 C).

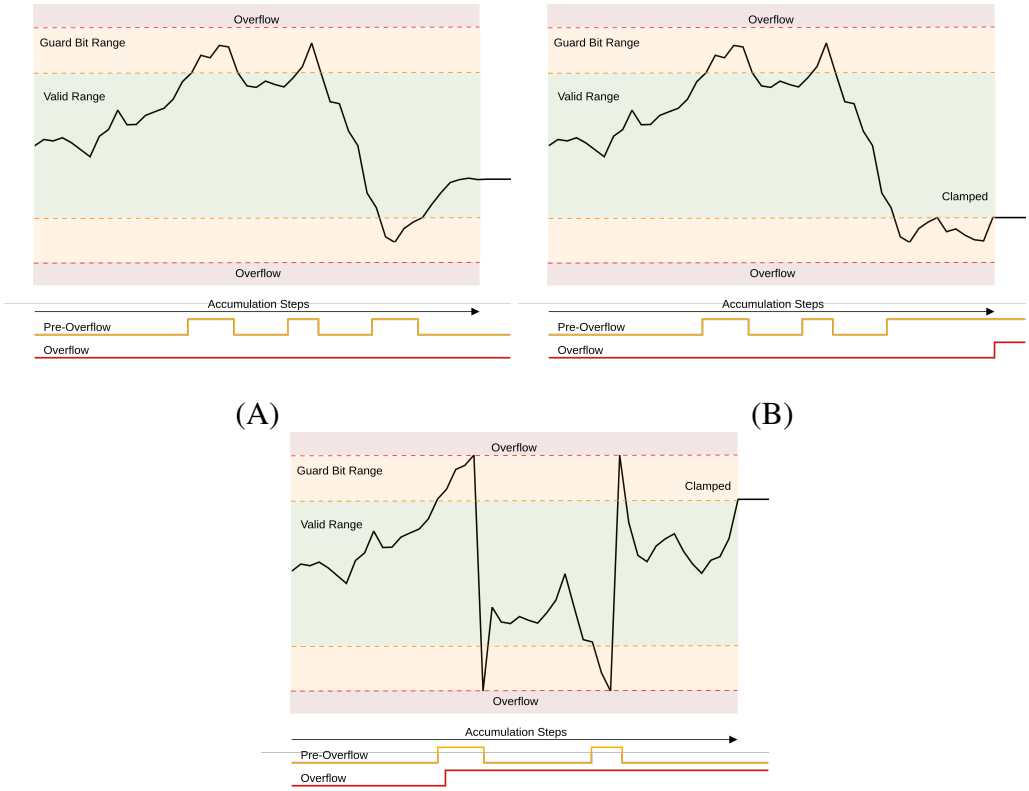
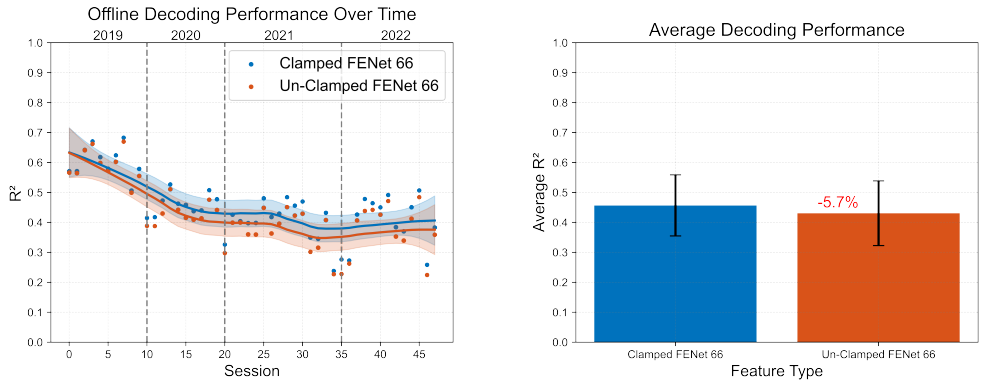



Figure 3.8: Accumulator Clamping Scenarios:
(A) Accumulator soft overflows but returns to valid range. No clamping occurs. (B) Accumulator soft overflows but does not return to valid range. Value clamped to maximum magnitude, in this case, negative. (C) Accumulator overflows and value rolls over. Clamping is performed such that the clamped value is clamped to maximum magnitude of the first overflow boundary.

The first scenario is ideal and means the guard bits prevented unnecessary clamping, and still allows for full representation of operation.The second situation is nominal, easily detectable with the values of the guard bits, sign, and carry values. The third situation is catastrophic if not properly handled as it could introduce an error equal to the entire bit range of the accumulator, which if it occurs within the first layer's traversal feature, may propagate to affect all features.

Since values from the accumulation register propagate through the entire network, a single overflow can vastly influence output features. In Figure 3.9, we show the simulated performance of FENet-66 operating on 30 kSps data with and without clamping. It is evident that without proper mitigation of overflow, the performance of the system is significantly degraded on average by 5.7%, with some days having a discrepancy of 17.5%.




(A) (B)

Figure 3.9: Clamping effect on performance of FENet-66 model on 30 kSps neural data: (A) Performance of FENet-66 clamped vs. unclamped. (B) Average performance over all sessions.

**Pooling Architecture**

The pooling block is responsible for storing the global average of each layer. Figure 3.10 details the pooling block architecture. To reduce the routing complexity of the processing element and reutilize the MAC hardware for the pooling register accumulator, as well as LReLU, and final accumulator division, the pooling block is implemented as a twenty-two-bit multi-precision-shift shift register. The shift hardware is equipt with multiple precision shift widths to allow a given operation both the flexibility to shift entire words, half words, or single bits to reduce the number of MAC clock cycles necessary to complete the pooling tasks while maintaining alignment precision. This is accomplished using the shift and insert hardware depicted in Figure 3.10 which shifts in the value in the pool reg shift in port based on the shift control.

The pooling block is also responsible for latching and exporting feature data such that the pooling register is free to begin the next computation while the feature value is exported. Upon the assertion of the capture feature signal, the last 9 bits of the pooling register are latched to the feature register. The control FSM ensures clock domain crossing safety by enforcing a full system clock cycle between any updates of the pooling register and latching of feature date. The feature shifting is accomplished through chaining the scan ports of active channels together. Inactivated channels or layers are multiplexed out of the scan chain.

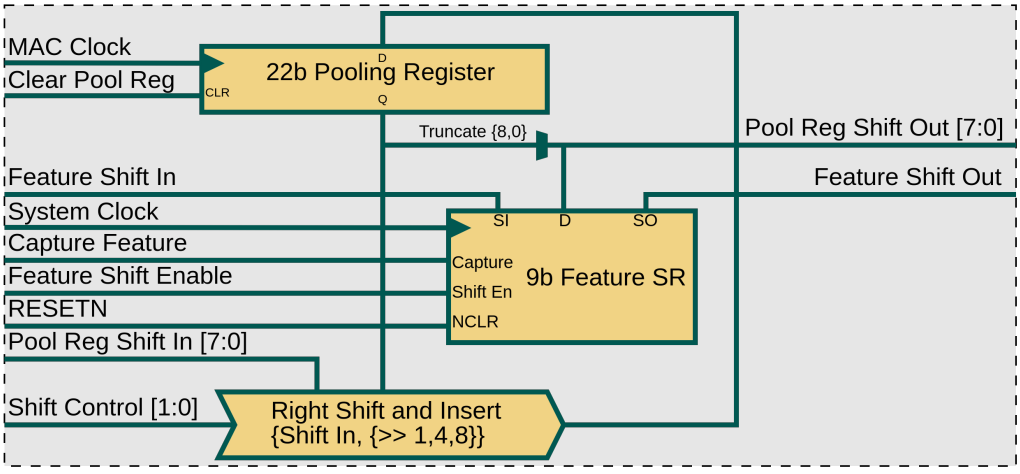



Figure 3.10: Architecture of pooling block including the 22b register, shift and insert hardware, and feature shift register.

**LReLU Sequence**

The center left branch of the processing element FSM is depicted in Figure 3.11 starting at the LReLU state and ending at the DONE state. To fuse the LReLU and pooling states, extra shifting cycles are added to the pooling shift and add sequence to allow channels individual shifting behavior dependent on the sign of their accumulator. Bit-precise shifting control is required to allow channels with either conditions to receive the same shift sequence, while performing separate behaviors. At the same time, performing the entire shift and add sequence one bit at a time incurs significant processing cycles which would increase the minimum processor clock frequency. To remedy this, variable shift and add capability is added to both the data path, and the pooling blocks. This minimal increase in complexity offers significant fused LReLU-pooling performance.

To reduce algorithmic complexity, the bottom half of the fractional bits are discarded by starting the accumulation sequence offset by six bits from the lsb. At the start of the ADD POOL state, negative channels are stalled by two bits, matching the number of bits configured in the model configuration registers. This allows positive channels to add their accumulation registers in full. Once the accumulator bits are skipped, both negative and positive channels continue with the pooling operation. During this phase, low bit precision is necessary, and the shift and add step is increased to a full word (8 bits). The LATCH POOL STALL state toggles the stall state of each channel such that the positive channels, which have finished their pooling operations, can stall pooling while the negative channels finish.

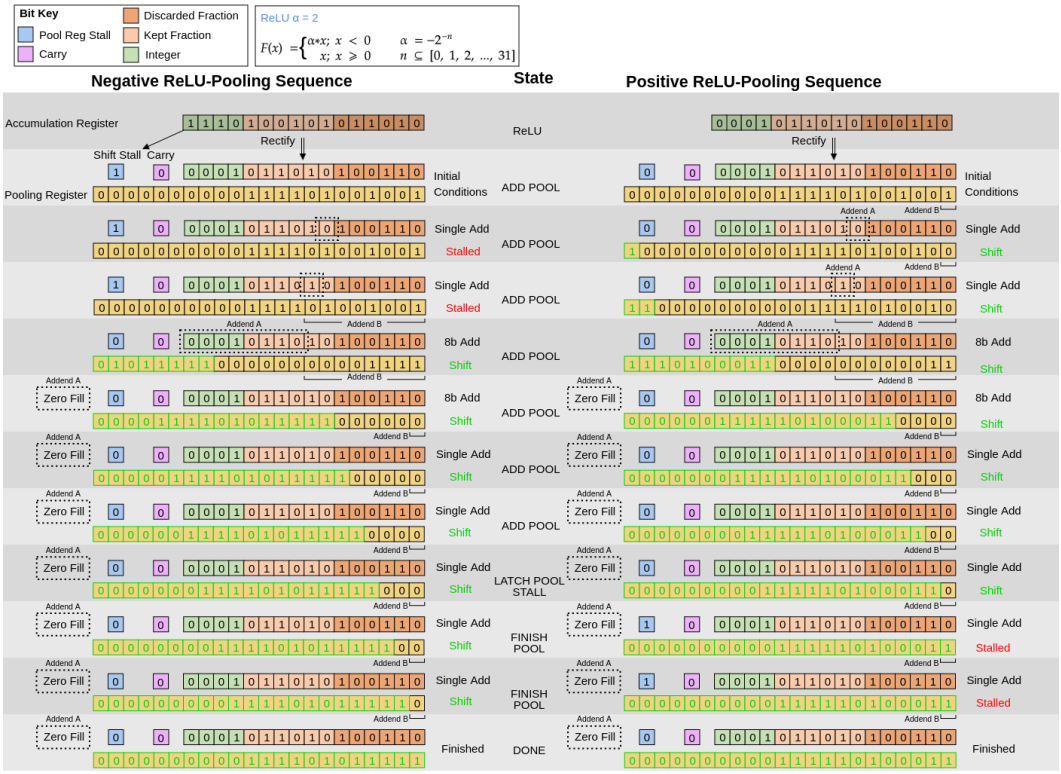



Figure 3.11: Fused pooling-LReLU sequence for positive (right) and negative (left) signs of the final convolution accumulation value. LReLU $\alpha$ parameter for this example is 2, meaning the accumulation register is shifted by 2 bits before addition to the pooling register.

**Layout Considerations**

In many cases of digital ASIC design, it is preferable to allow automatic place and route layout large swaths of the design so it can take advantage of large scale modeling for optimizations of logic placement, and sizing. However, with this system, granular channel and layer level gating of power was a high priority. As such, the layout needs careful customization to create the complex power delivery system. The primary layout constraint was the width of the processing element, which was constrained to $\frac{1}{8}^{th}$ the width of the SRAM, which worked out to be 50.8 $\mu m$ wide. The second constraint was that the pin layout for all processing element IO is to be on the top edge of the design so that the centralized control signals could be distributed from an always on domain (located above the processing element). The processing element layout is depicted in Figure 3.12. The MAC data paths are the most active components of the processing element, so they are placed as close as possible to the IO ports of the MACRO. In a similar fashion, each layer is situated such that the most active layer is closer to the MACs.

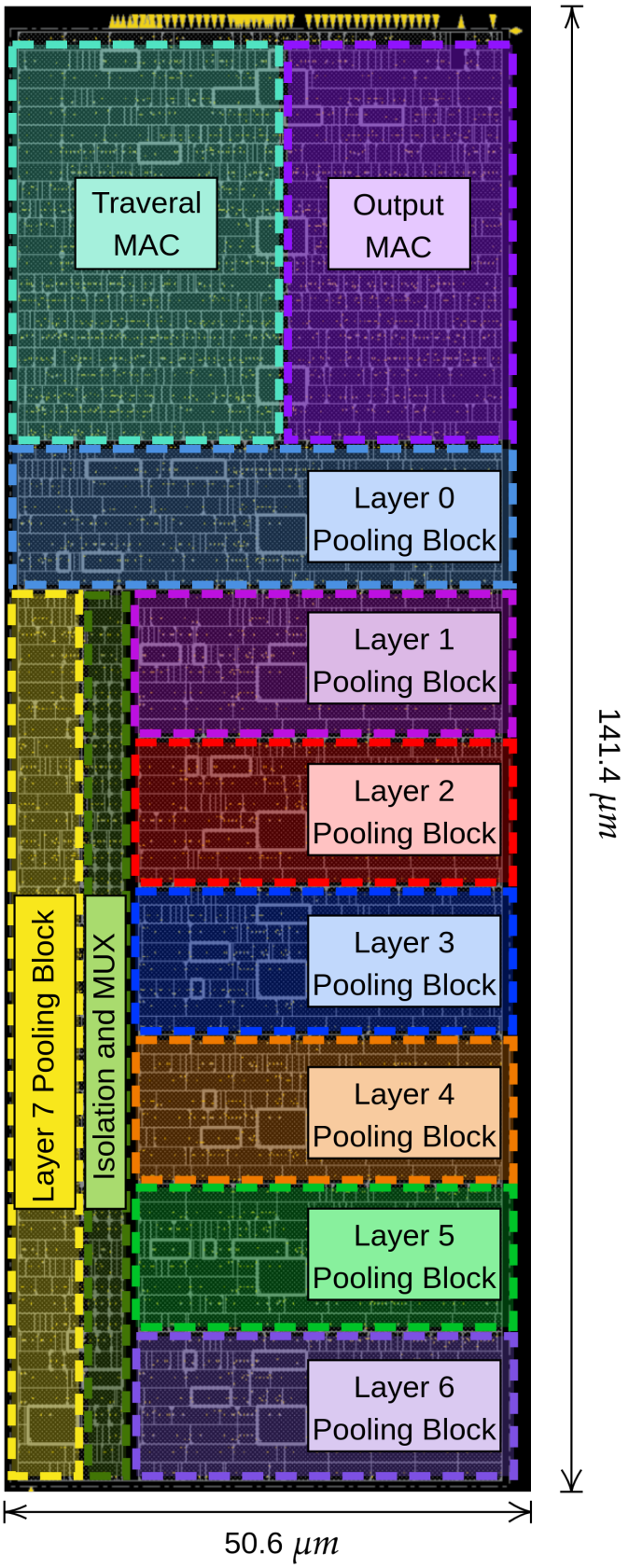



Figure 3.12: Layout of processing element with each major component group identified.

### Power Gating

Power switches run the spine of the processing element. Each processing element is power switchable such that inactive processing elements can be de-powered to maximize efficiency when few channels are needed for a given task. Furthermore, models with fewer layers leave these blocks unused. The static power draw of each channel can be up to 30% the total power consumed by the processing element, especially when operating with small models at low frequencies. Mitigating these costs allows for high linearity in the power scaling of the system. Incorporating such a complex power distribution system allows the architecture to shed the static power costs incurred by the hardware needed to support larger models, when processing models with fewer layers.

One important consideration to make when implementing power gating is the voltage headroom of the incorporated logic. All of the processing element logic is chosen to be LVT such that its minimum operating voltage can be pushed as low as possible, thus conserving dynamic power. Using HVT power gating cells is common because they provide the lowest leakage when the cells are turned off. When designing a system which is intended to operate near the threshold of the CMOS logic, HVT power gating cells will enter the linear operation regime at a higher voltage than LVT cells, increasing the required VDD of the system overall. Figure 3.13 shows the delay of the maximum path of the processing element versus VDD. For a delay margin of 5 nS, the minimum operating voltage is shown to increase by .05V when using HVT power gating cells. To mitigate this, standard cell power gating transistors were modified to use LVT transistors, improving the minimum operating voltage of the processing element by .05V.

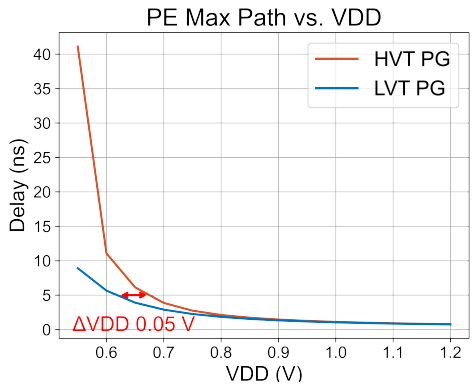



Figure 3.13: Effect of threshold voltage on the delay of maximum path over VDD.

This modification comes at the cost of 200× increased leakage current when the channel is powered off. That said, this off-state leakage current was simulated to only increase from 10 $nW$ to 2 $\mu W$ over the entire chip. This cost is far outweighed by the reduction in VDD, since the total change processing element power going from 0.63 $V$ to 0.68 $V$ can exceed 50 $\mu W$ (measured using FENet 240 at 400 kHz). See Section 5.2 for the implementation results.

## 3.3 Channel Block Macro

This section details the architectural design of the channel block macro. First, the overall composition of the block will be discussed, which is then followed by the architecture and design decisions made for several important components of the block such as the SRAM, asynchronous queues, and signal feed-through buffers.

### Channel Block Overview

Figure 3.14 provides a schematic overview of the channel block with a focus on the association of channels with their memory, and the division of power domains. Each channel is associated with an eighth of the SRAM bandwidth. Input data is temporarily cached in an asynchronous FIFO before being written into the first layer of SRAM. The memory (MEM) and processing element (PE) voltage domains are interfaced with level shifters.

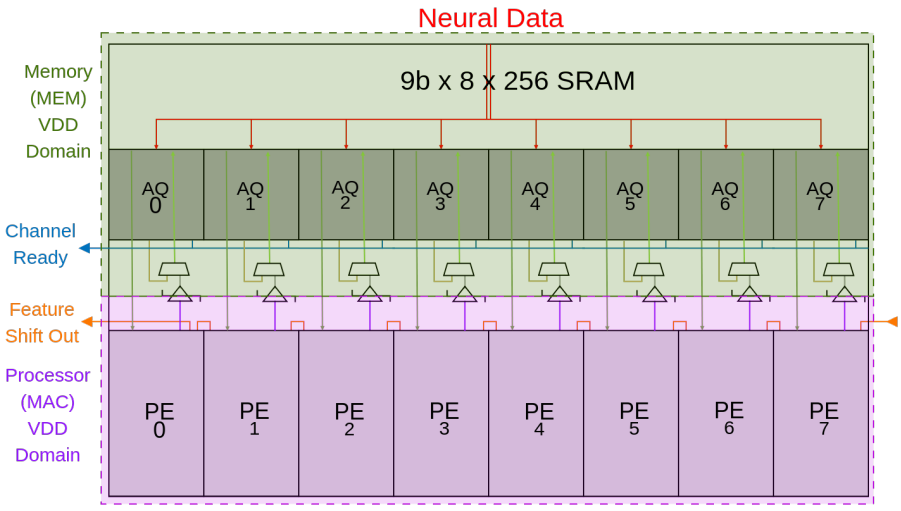



Figure 3.14: Schematic overview of a single channel block focusing on the associations of a set of channels and their SRAM. Division of power domains are depicted with shaded boxes with green shading indicating the higher voltage MEM power domain, and the purple shading indicating the low voltage processing element domain.

To support system scalability, the processing hardware and activation/feature caching are integrated into a modular channel block macro. Each channel block contains 8 processing elements (PEs), each with their own asynchronous data queue, gated control-signal buffers, and power-domain level shifters. All PEs share a customized TSMC 72-bit $\times$ 256-element low-leakage single-port SRAM macro.

To accommodate the higher-voltage headroom requirements of the proprietary SRAM, we separate the memory and processing element VDD domains. Level shifters, shown in Figure 3.1 Block 0, are inserted only where necessary—specifically at the interface between the 9-bit intermediate feature outputs and the SRAM write port multiplexer.

We designed the physical layout to ensure that each PE occupies one-eighth the width of the SRAM macro, minimizing wire length between memory and compute units. This tight spatial coupling reduces energy consumption and facilitates tiling of multiple channel blocks across the die for scalable expansion.

Control signals and weight data, originating from centralized FSMs, are buffered using balanced AND gates. These buffers are deliberately timed to create a small positive skew relative to the system clock, preventing hold violations as the chain of channel blocks grows. Buffer placement and sizing are optimized to minimize both inter-signal skew and clock-relative skew, maintaining signal integrity even in large system configurations.

Each channel includes an asynchronous queue that interfaces with the validation data stream. The asynchronous design allows data generators to operate on independent clocks relative to the system and processor element, significantly relaxing timing requirements on upstream acquisition hardware and improving system scalability.

**Activation Memory**

In deep learning processors, data movement and storage consumes the majority of power for the system. At a simple glance, it may seem that all one needs to do is pick the amount of SRAM needed for a task, and compile an SRAM block which meets those specifications. However, there are far more factors to consider. One such factor is how data is organized and partitioned within SRAM, and can significantly affect the mapping of processing elements onto the workload. Another factor is where the data will need to travel after it is read from the SRAM. If SRAM is significantly separated far from processors, efficiency will be lost to power necessary to buffer these signals.

To optimize an architecture for the streaming FENet workload, the partitioning method was considered in detail to reduce the necessary SRAM size in bits. Once a minimum SRAM requirement was determined, the sizing of each SRAM was explored to determine the optimal aspect ratio to minimize power required from the system.

### Partitioning

The memory partitioning was designed with the worst case requirements as determined by the algorithmic exploration discussed in Section 4. These requirements were derived from the largest necessary model (FENet-240) with the largest necessary data bin (900 samples). Conventional architectures require all data for a workload are present during processing such that activation reuse can be optimal, resulting in the partitioning scheme depicted on the left side of Figure 3.15. In the neural decoding environment, however, data is streamed slowly as it is recorded from the multi-electrode array. Conventional architectures and mapping schemes would therefore require caching of an entire bin of neural data. With 900 (9 bit) samples per channels and 192 channels, caching the activations alone would require 1.56 Mb of SRAM without even accounting for the space required for intermediate activation scratch space.

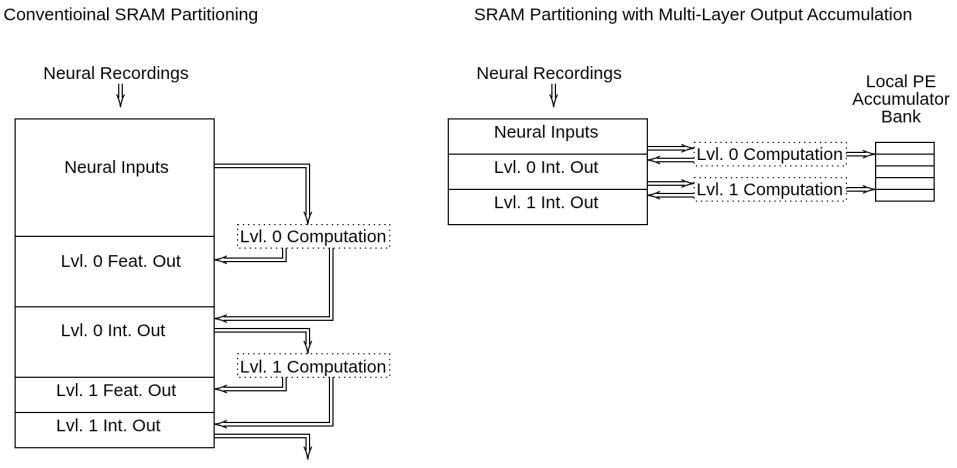



Figure 3.15: Comparison of the memory spaces for conventional SRAM partitioning schemes and those using local partial sum accumulators in their processing elements.

To mitigate this requirement, the partitioning scheme shown on the right of Figure 3.15 was devised to take advantage of the streaming nature of the neural data. Since each sample arrives at a relatively slow rate ($\approx \frac{1\ sample}{33.3\ \mu s}$), computations are scheduled in-between each sample's arrival. The minimum number of SRAM elements for a given model therefore reduces to the maximum 'history' needed for any computation, which becomes possible after any 1 sample is received. This worst-case scenario occurs when the arriving sample triggers convolutions for all layers. The maximum SRAM depth is therefore just the sum total of all pass kernel weights. Since SRAM is optimally addressed in powers of two, a max kernel depth of 256 was chosen so that it can fit the largest FENet models. Notably, this does not fit the model from [22], however, results from model reduction in retraining in Section 4.1 showed that the negligible difference in decoding performance would not ameliorate the increase in complexity, power and area.

**SRAM Sizing**

Figure 3.16 depicts the range of shapes and sizes of which memory can take, ranging from a single SRAM block that contains all necessary memory space that supports only one read/write port for all processing channels, to very shallow memory with individual ports for each channel. The ratio between the depths (in number of elements) and width (number of supporting channels) is known as the aspect ratio. The aspect ratio of the memory has a significant impact on access power. Deep, narrow memories both require more accesses to curry data to and from processors, and have much a large bit line capacitance. However, when these activations can be reused, the additional power from SRAM access can be outweighed by fewer independent data channels that pass activations in-between processing elements.

To optimize the memory structure for FENet, two criteria were considered. The first is the reuse nature of the FENet workload, and the second is the energy for each access. To understand the optimal aspect ratio a system architect must determine which aspect ratio best suits their data network. Workloads with large levels of reuse are able to locally pass activations to neighboring processing elements. FENet, however, has very little activation reuse, therefore, this architecture reduces data travel by tightly coupling SRAM with processing elements. Using a minimal SRAM partitioning scheme informs the memory layout design by determining the minimum block size necessary for a given channel.

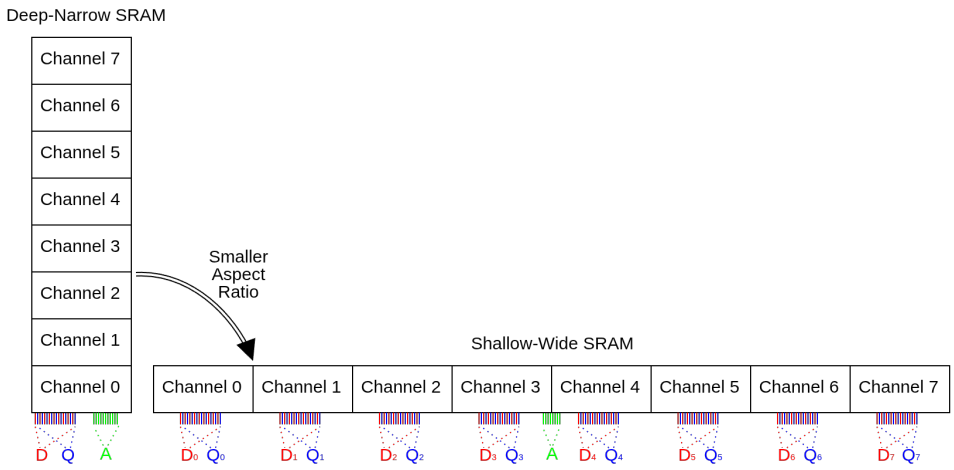



Figure 3.16: Range of SRAM aspect ratios depicting the physical difference between narrow-deep SRAM and wide-shallow SRAM and how the memory spaces would be distributed within a single SRAM instance. A, D, and Q are the address, input, and output ports of the SRAM, respectively.

For the FENet memory requirements (256 elements × 9 bits × 192 channels), the SRAM power is estimated using the relation in equation (3.1):

$$\begin{aligned} N_{SRAM} &= \left\lceil \frac{Total\ Words}{W \times D} \right\rceil \\ AR &= \frac{D}{(W \times N_{SRAM})} \\ F_{access} &= \frac{N_{ch.}}{N_{SRAM} \times W} \end{aligned} \tag{3.1}$$

$$P_{SRAM} = N_{SRAM} \times ((P_R \times N_R + P_W \times N_R) \times F_{access} \times FPS + P_L)$$

where $Total\ Words$ defines the total number of words required for the FENet system, $W$ and $D$ are the width and depth of an individual SRAM block in number of words. These parameters are used to determine the total number of individual SRAM blocks required to fit the entire workload ($N_{SRAM}$).

For a given choice of SRAM size, the total aspect ratio of the memory system ($AR$) is then calculated. The bandwidth ($W \times N_{SRAM}$) then determines the number of accesses for a given block necessary for all transfers ($F_{access}$). $N_R$ and $N_W$ are the number of SRAM reads and writes for a given workload and are related to the number of MACs and pooling operations, defined by equations 3.2:

$$\begin{aligned} N_R &= \sum_{l=0}^{N_L} \frac{M_l}{2} \\ N_W &= \sum_{l=0}^{N_L} B_L \end{aligned} \tag{3.2}$$

where $N_L$ is the number of layers, $M_l$ is the number of MACs per layer, and $B_L$ is the bin length for each layer. Specifically, $N_R$ will be half the number of non-padding macs, and $N_W$ will be the size of each layers inputs.

The SRAM read, write, and leakage power ($P_R$, $P_W$, $P_L$) are then determined from SRAM data sheets for various combinations of $W$ and $D$ which are used to finally estimate SRAM power ($P_{SRAM}$). Using SRAM access activity for FENet-66 with 150 sample bin sizes (5 kSps with 30 ms bins), the SRAM power with respect to the aspect ratio is plotted in Figure 3.17. With this simulation, it becomes obvious that minimal SRAM power is achieved with the smallest aspect ratio. These observations resulted in the selection of 24 SRAM blocks sized at 256 elements deep and 8 words wide, informing the total block size of the channel block macro.

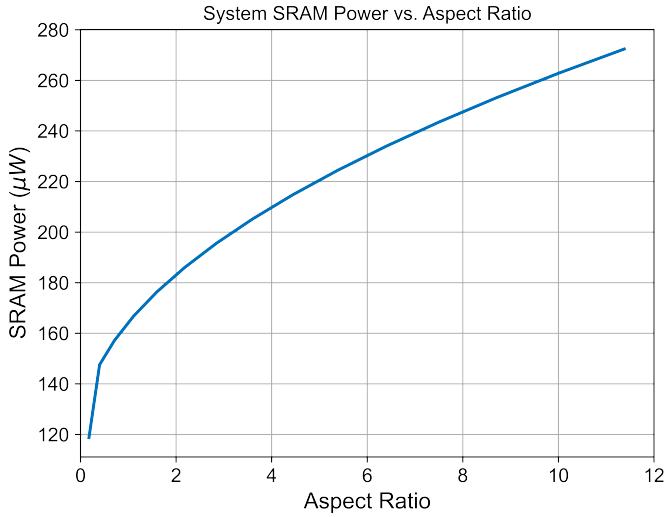



Figure 3.17: SRAM power for the FENet workload with respect to aspect ratio.

**Asynchronous Queues**

Asynchronous input queues are included for each channel to provide storage for activations when SRAM is busy with processing and to buffer data while all channels are being filled. Crossing asynchronous clock domains is challenging because data edges can coincide with the receiving register's clock edge, increasing metastability risk. To enable reliable transfer, we implement queues in which write events to a first-in, first-out (FIFO) buffer in the write-clock domain are registered into the read-clock domain with synchronizers. The read domain then uses these events to determine when it is safe to capture data from the FIFO. The FIFO architecture used in our design is shown in Figure 3.18. To ensure correct operation across domains, the read and write pointers use gray code and are timed to meet a maximum path delay equal to the fastest clock period. An extra bit in the gray-coded address prevents the write pointer from overrunning the read pointer when the buffer is full. These buffers interact with both the write and read domains through *full* and *data-available* flags, respectively. These queue signals are depicted as the signals color coded orange in Figure 3.1 Block 1 and are aggregated across channels using simple reductive logic to nullify the affect of de-powered channels.

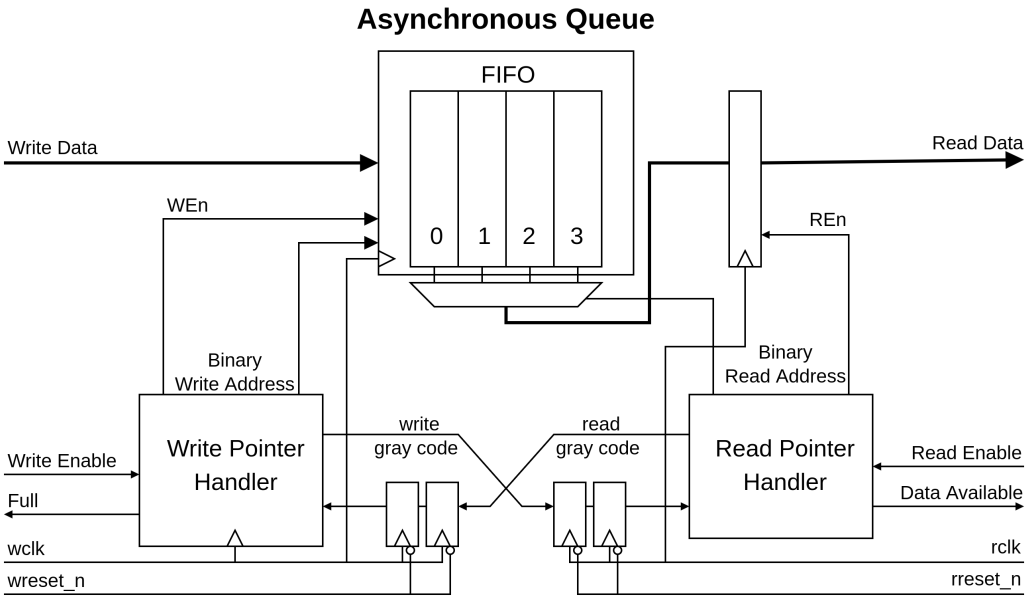



Figure 3.18: Schematic Diagram of asynchronous queue. Write and read pointers are synchronized before used in handler logic.

**Control Feed-through Buffers**

The feed through buffers serve to broadcast the control signals from the control finite state machines, or the prior channel block in the street, to the next block in the sequence. Each buffer is composed with a skew-optimized AND gate, commonly used within clock gate structures. The logical position of the feed-through buffers in relation to the objects they drive is shown in Figure 3.19.

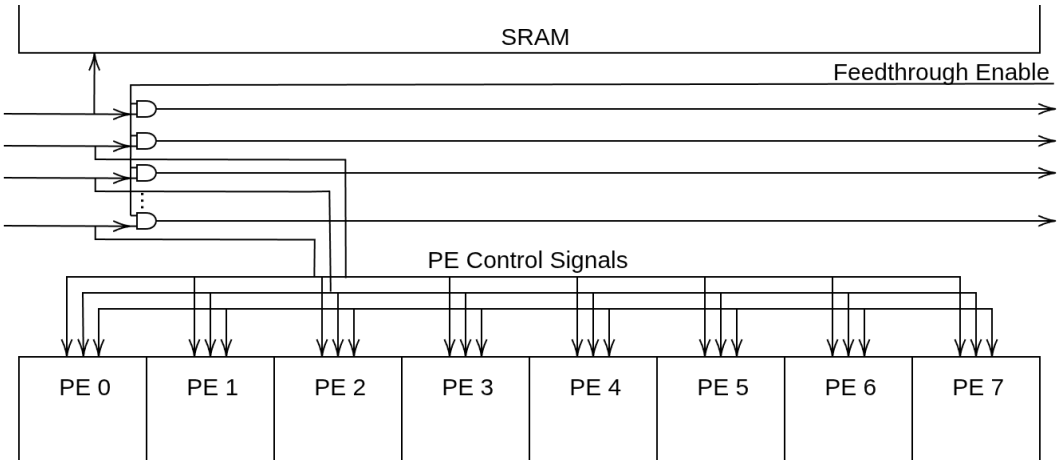
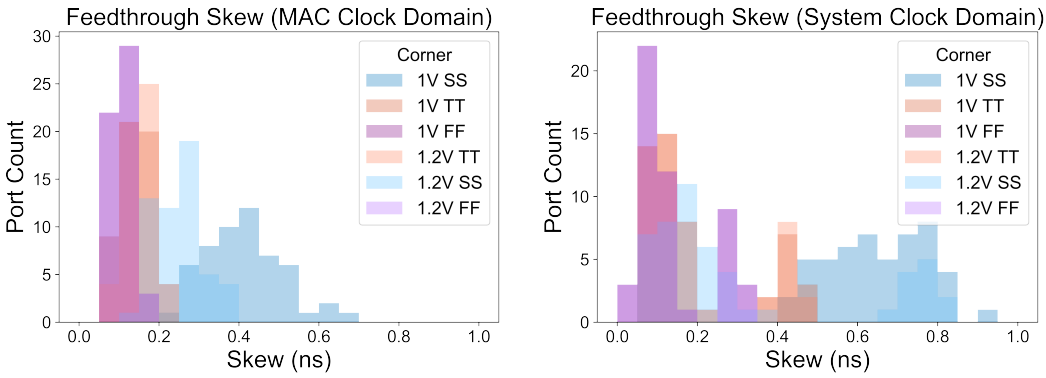



Figure 3.19: Depiction of feed-through buffers in relation to the logic that are driven by them. The control signals are gated with the feed-through enable signal which is asserted if any channels in the street above this block are enabled.

Effectively broadcasting control signals to large numbers of processors is non-trivial as signal skew accumulates between control and clock signals. To mitigate these issues, signal buffers were implemented with skew-balanced clock buffers, and timed such that distributed signals have a minimal, defined skew, which is longer than those for the system and mac clocks to prevent hold violations. The buffers are also co-located within a tightly defined region within the channel block macro to minimize spatial variation. These measures ensure that only positive clock skew is imparted from one channel block to another. Figure 3.20 displays the distribution of skews for each clock domain which is fed-through the channel block MACRO. The processing element has the tightest timing constraints, and so its skew will have the largest influence on the maximum operating frequency. Even in the worst case, skew is only 0.7 ns per block. For a 4 block chain (supporting 32 channels), this only imparts a skew of 2.1 ns, which is acceptable for the expected operation frequency. Since clock and signal distribution is a large portion of the system power consumption, feed-through buffers are implemented as AND gates; enabling the mitigation of switching power when entire blocks in the chain are disabled.




(A) (B)

Figure 3.20: Skew Distribution of Control Feed-through Signals:
(A) Processing element (MAC) clock domain. (B) System clock domain.

### Level-Shifters

The use of low leakage SRAM enables high efficiencies of memory-intensive algorithms at low speeds by mitigating leakage power, but inherently increases the operating voltage of the memory domain. Separating the processing element VDD domain allows low-power operation, however requires level shifting of data from the low voltage processing domain to the high voltage memory domain. This is achieved with low-to-high level shifters embedded within each asynchronous queue.

## 3.4 Control Hardware

This section will discuss the algorithm design of the control hardware which directs the operation of the processing hardware. These blocks were designed as finite state machines and shared globally among all of the processing components. Responsibilities of these control blocks range from generating SRAM addresses for reading and writing into the SRAM storage cache, to generating control signals that are consumed by the processing elements to enable certain functionality.

An overview of the hardware used to generate the control signals and state behavior is shown im Figure 3.21. Each layer has a dedicated state machine defining which state the layer is currently occupying. Using this state information, the scheduler updates pointers for memory control, which in turn update the layer's state information. The processing element finite state machine is also influenced by the state of the active FSM state. This state indicates which operation necessary for this control cycle. For instance, during the convolution CNN control state, the processing element FSM will generate a control sequence for the processing element that results in a multiply-accumulate operation, while during the update state, the processing element FSM will instead generate the controls which result in the addition of the accumulation register to the active layer's pooling register. The following sections provide more detail about the behavior of the control finite state machine behavior.

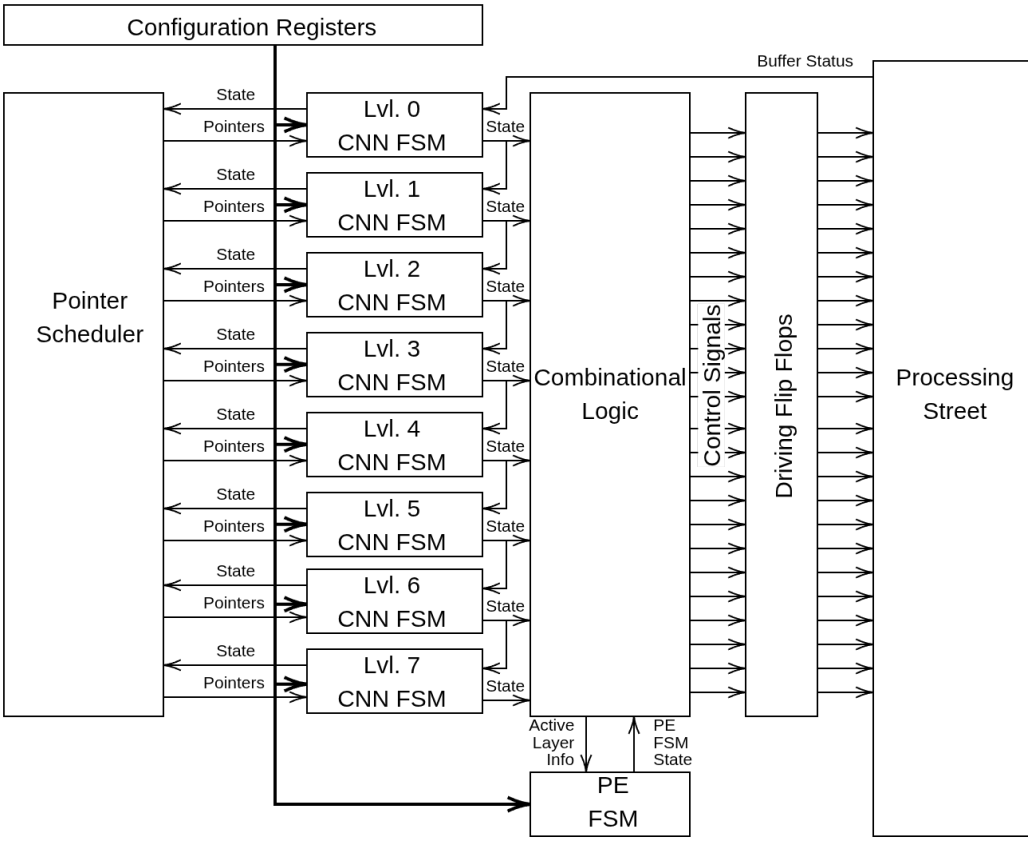



Figure 3.21: Depiction of the hardware defining the control finite state machines and their relation to one another. Each layer has a dedicated finite state machine which is also dependent on the states of the other layers to determine if it is their turn to change state or not. Control signals are generated from these state machines and synchronized to the rising edge of the clock before they are broadcast to the processing hardware.

### CNN Control

Operating on streamed neural data imposes unique constraints on CNN architectures. At 5 kSps sampling, new activations arrive every 200 µs, meaning that only a fraction of the data needed for a full convolution layer is available at any given time. In contrast, conventional CNN accelerators, particularly those designed for 2D image processing, typically assume that input data can be accessed from local memory when needed. For streaming neural interfaces, caching the entire activation history would be impractical in terms of memory and resource cost, motivating an architecture that processes data in time as it arrives.

We implemented a streaming-oriented CNN control FSM to generate SRAM addresses and sequence layer operations so that higher-order layers only compute once sufficient data (one stride) is available and processing element resources are free. The algorithm also manages efficient zero-padding by dynamically adjusting kernel width: growing at startup, maintaining a constant size during steady-state processing, and shrinking at completion as the active window exits the kernel. Figure 3.22 illustrates this control across three layers during startup, steady-state, and completion phases. It also depicts the interaction between states of each layer. These interactions include the triggering of the loading state of higher-order layers by the completion of a lower-order layer's convolution and higher-order layer's priority over MAC resources to free up their memory spaces for new intermediate activations from lower-order layers.

The stall data signal in Figure 3.22 indicates when the asynchronous queue (AQ) is full and to apply back-pressure on the data sources. During the conclusional padding state (Figure 3.22 C), memory resources become busy, preventing the loading of new data. By optimizing the system frequency to consume data at the optimal rate for a given model, back pressure is minimized or eliminated altogether.

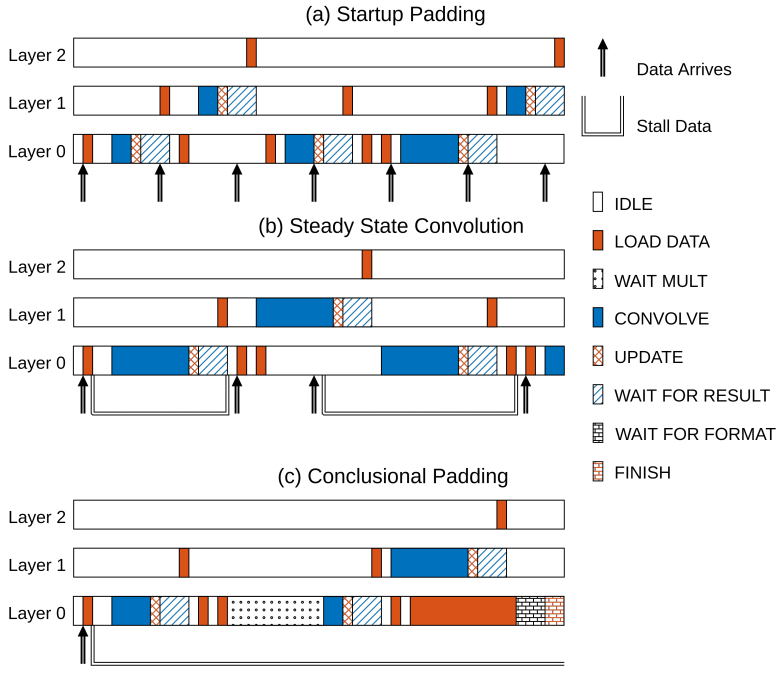



Figure 3.22: Control behavior for 3 layers of the CNN depicting the 3 padding state behaviors: (a) Startup Padding (b) Steady State Convolution (c) Conclusional Padding.

The CNN sequence is directed by a group of centralized FSMs (one for each layer), which ensure that the states of each layer are compatible with each other to prevent resource contention. Each layer will wait in the `IDLE` state while higher order layers complete their computations, which frees up memory within the higher order layer's partition. The `LOAD DATA` state writes activations to a layer's partition, and is triggered either by partition space becoming available (layer 0), or a lower order layer finishing a convolution (layers 1-6). On the completion of loading a full stride of activations into a layer's partition, the layer enters the `CONVOLVE` state, on the condition that memory space is available in the higher-order layer and processing element resources are available. If neither one of those two criteria are met, this layer is stalled by waiting in the `WAIT MULT` state. Following the convolution, the `UPDATE` state adjusts the convolution pointers for the next convolution. In the `WAIT FOR RESULT` state, LReLU is applied and partial sums are either added to the pooling register (when generating an output feature), or to the higher order layer's SRAM partition (traversal path for layers lower than the last layer). Finally, the `WAIT FOR FORMAT` state normalizes and rounds the feature in the pooling register to 9 bits. The `FINISH` state waits for higher order layers to finish their padding computations before commencing a new sequence.

**Scheduling**

Efficiently mapping resources to a given CNN algorithm determines the utilization of hardware resources, and as a result, the efficiency and performance of the system. Furthermore, in order to allow full flexibility to all FENet model architectures while using all memory resources as efficiently as possible, the scheduler should allow for arbitrary kernel widths for each layer. Efficiently processing stream data requires algorithms built to process data as it is received. The recursive pyramid algorithm was developed for processing discrete wavelet transforms and avoids processing data bins in batches [59], reducing the required memory space for a processing bin of N elements from O(L Log[N-1]) to just O(L), where L is the total length of the wavelet filter. We apply this algorithm to the processing of streaming convolutional neural networks to make similar memory utilization improvements where the required memory space is reduced to the total sum of all kernel widths. An overview of the implemented scheduling algorithm is depicted in Figure 3.23.

The figure depicts the various memory pointers used within a given layer's memory space, and the corresponding decision tree that determines how the pointers should be updated. The dark teal boxes on the left represent individual memory addresses within SRAM for a given layer. The boxes above and below are the N + 1 and N - 1 layers, respectively. Each layer has a dedicated set of pointers which mark important boundaries within a memory space. Colored pink, the stand and end boundary pointers define the boundaries which the layer's SRAM extends within the entire SRAM block. These pointers are bound by the values defined within the configuration registers, however they grow and shrink at the start and end of convolutions as is discussed the the next subsection about the padding algorithm. The convolution pointer in black defines the current memory address. The blue pointer defines the start and end points of the convolution sequence. A convolution sequence will start at the front pointer, and wrap around at the boundary pointers until they reach the front pointer once again. The decision tree on the left defines how these pointers are updated based of the various states of the CNN finite state machines.

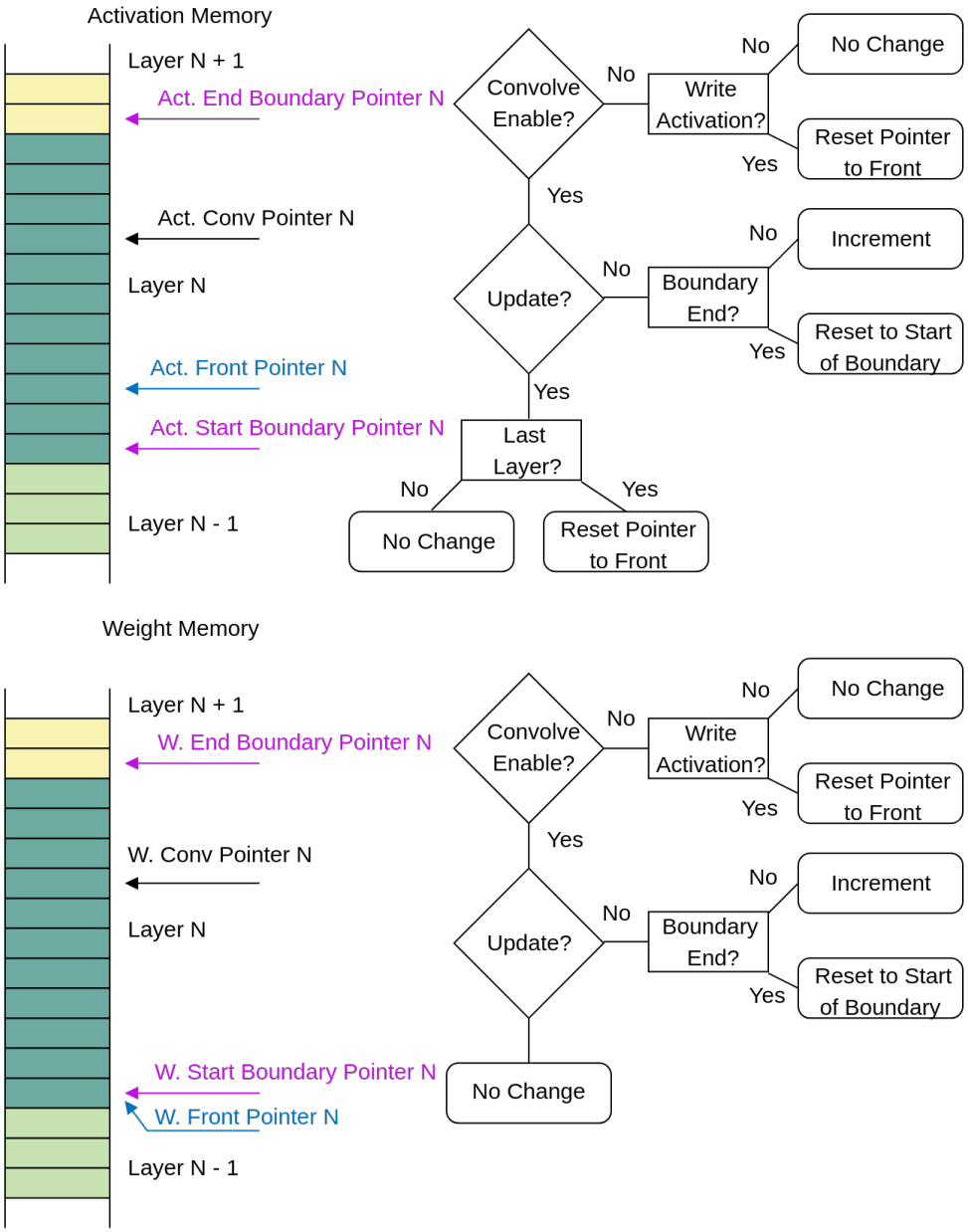
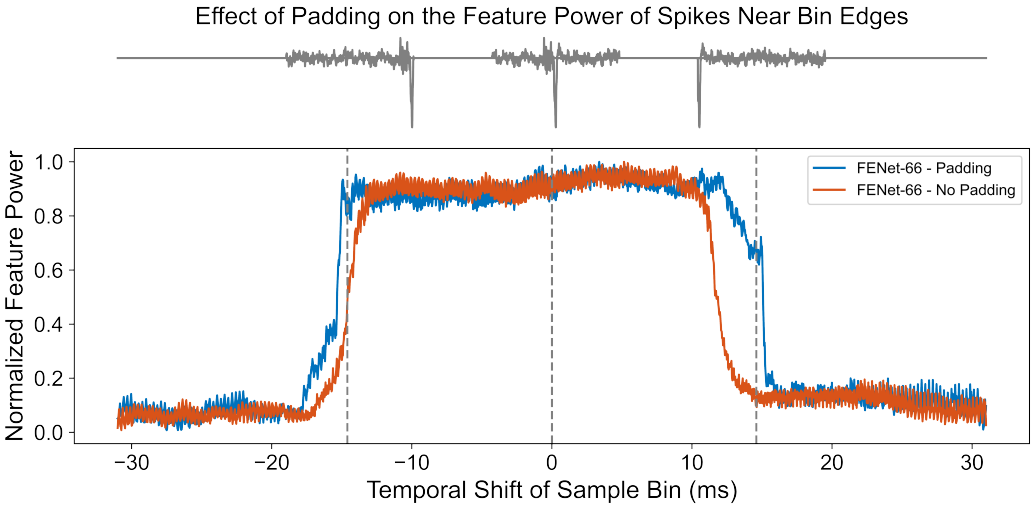


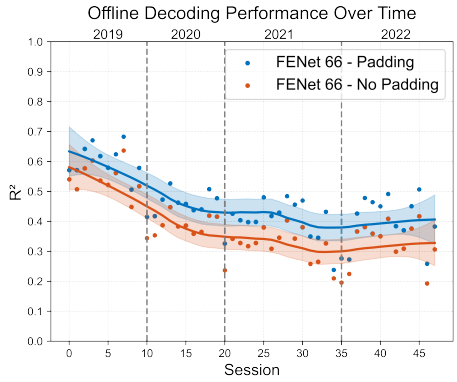

Figure 3.23: Diagram of the scheduling algorithm for activation and weight SRAM Access.

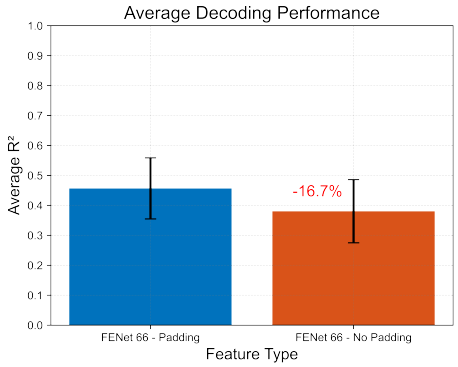
**Padding Algorithm**

Padding in convolutional neural networks helps preserve information that is present at the edges of sample bins. Without padding, the feature extraction will develop blind spots where it is unable to extract neural data at these indices [2]. Unfortunately, padding also introduces extra complexity. In Figure 3.24, the performance of FENet-66 operating on 5 kSps data is shown with and without zero padding. Figure 3.24 A shows a single neural spike waveform set in noise. As the spike approaches the edges of the sample bin, the model without padding is unable to detect it. This leads to a 16.7% reduction in feature performance as is observed in Figure 3.24 B and C.




(A)




(B) (C)

Figure 3.24: (Effect of zero padding on decoding performance: (A) Feature Power as a single spike is offset within a neural data bin. (B) Effect of padding over 48 sessions. (C) Average $R^2$ performance of padded and non-padded models.

A rough estimate of the number of padding convolution cycles is shown in equation (3.3):

$$Padding\ Cycles = \left\lceil \frac{K_L - S_L}{S_L} \right\rceil \tag{3.3}$$

where $K_L$ and $S_L$ are the kernel width and stride of the $L^{th}$ layer, respectively. Somewhat troubling is the fact that each of these padding cycles themselves produce an intermediate output which is passed to the $L+1$ layer. The conclusional padding, therefore, does not start until the lower-order layers finishes its entire padding sequence. A naive approach to this design challenge is to simply fill the activation memory space with zeros, and carry on with computation as normal. However, with large FENet models like FENet-240, zero padding can comprise nearly 27% of the total required MAC operations, and therefore it is paramount that an efficient algorithm be developed to skip these unnecessary zero multiply operations.

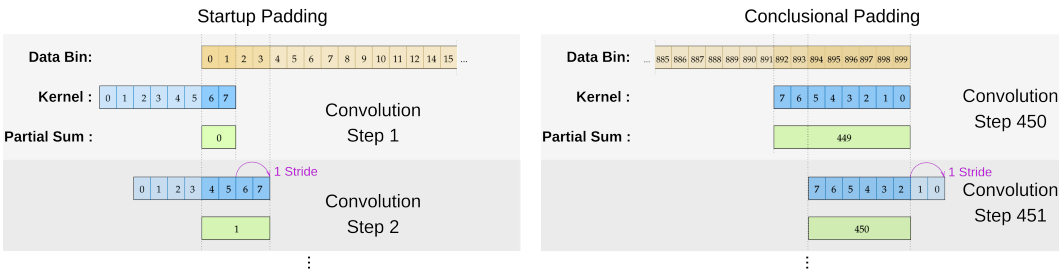
The solution developed for this design challenge was to introduce finite state machines for each CNN layer and were briefly mentioned in Figure 3.22 and Section 3.4. These state machines take one of 4 states: `Startup Padding`, `Steady State Convolution`, and `Conclusional Padding`, and `Finished Padding`.

### Startup Padding

The padding during the initial stage of computation requires the weight vector start and end pointers to grow as more non-padded neural data is streamed into the system. A single step in padding is depicted in Figure 3.25 A. A new convolution does not begin until a full stride of data is received. At this point, a convolution commences, with the only the non-zero-padded regions computed. The start address for weight retrieval is decremented until it reaches the bottom most address of the kernel's memory field, at which point the system enters steady state operation.

### Conclusional Padding

Once a full bin of neural data is received, the layers enter the conclusional padding state. In this state, the end pointer of the weight vector is decremented until the full kernel is covered. Each layer only enters its conclusional padding state once the previous layer has finished its own conclusional padding. This sequence is depicted in Figure 3.25 B and shows how the active kernel region shrinks to facilitate zero padding without actually completing the zero multiply operations.




(A) (B)

Figure 3.25: Padding Behavior:
(A) Behavior of system as a new bin of neural data is streamed into and the system. (B) Behavior of the system as the streamed data reaches the end of a convolution bin.

### Edge Case Behavior

A special edge case occurs in padding when a layer (often higher-order layers) does not receive enough data to exit the startup padding state. In this case, both the start boundary and the end boundary pointers are decremented until the patch of valid neural data is fully convolved across the entire kernel.

### Model Flexibility

This architecture maps a wide range of 1D CNNs by modifying only the FSM sequence (kernel, stride, LReLU $\alpha$, and bin width), avoiding hardware redesign or multi-step data remapping. It can generate 2–8 features from up to seven feature-producing layers plus one terminal traversal feature. Kernel width is limited only by the 256-element weight SRAM (e.g., one 256-tap layer or two 128-tap layers). Stride is unconstrained within that depth. LReLU leak parameter ($\alpha$) is a configurable right shift of 0–32 bits. Similarly, the pooling-register division can be configured with a 1–32-bit shift. Bin width is set by the product of the first-layer stride and a cycle-count parameter, which has a maximum value of up to 2048 cycles. To ease timing, a start delay of up to 32 MAC clock cycles after the system-clock rising edge is configurable. Together, these settings expose a large hyperparameter space for 1D CNN applications while keeping hardware mapping straightforward.

## 3.5 Processing Element Control

The design ethos of this processing element was to minimize overhead, while still handling all the tasks of the FENet workload. In a conventional design, the multiply-accumulate or MAC unit is synthesized directly with full multipliers and adders. This allows each operation to finish in a single clock cycle, which is great for high-throughput systems. Neural data on the other hand doesn't arrive that fast — it comes as a slow stream. So all of that extra circuitry for one-cycle execution becomes overkill. Instead, this architecture takes advantage of the long periods between data arrivals. Rather than doing everything in one step, I can break the operations into smaller pieces, spread across several cycles using much simpler hardware.

Figure 3.26 depicts how a single multiply-accumulate operation is broken up into sub-phases of the system clock. Each shift of the activation data is accompanied with a single bit of each weight for both the traversal and feature generating paths. Each bit receives two processing element clock periods to allow all 16 bits of the accumulation register to complete their addition. All operations that the processing element is responsible for are broken down into elementary shift and add components, and fit into a single system clock period.

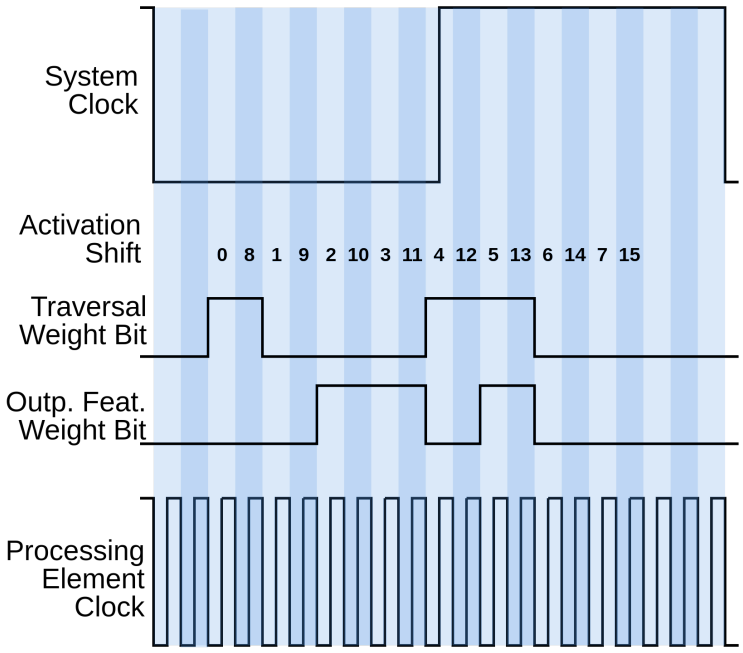



Figure 3.26: Time-multiplexed phases of multiply-accumulate operation within the period of a single system clock cycle.

Generation of control signals for the processing element for each of these processing phases is performed with a finite state machine with the behavior depicted in Figure 3.27. The functions of these control signals include directing the barrel shifters to align data for the 8 bit adder, directing the negation of the operands to the adder, controlling the shifting of the pooling registers during pooling, global approximate averaging, LReLU, rounding, and quantizing, as well as serializing the weight. To save power and area, the state value was encoded as one-hot, such that the state value requires little decoding when interpreted by channel logic, and control signals were latched while weight bits are 0 to reduce switching activity.

The FSM state is mainly determined by the state of two other minor state machines. One of these state machines is the accumulator barrel shift control, determining the size and offset of the active region, and the pooling register shift count, which keeps track of how many bits the pooling register has shifted. These minor state machines have behavior that depends on the state of this FSM, which in turn waits for those minor state machines to finish their sequence before continuing with its own sequence. This set of tasks are accomplished using thirteen states with the state relations and responsibilities shown in Figure 3.27.

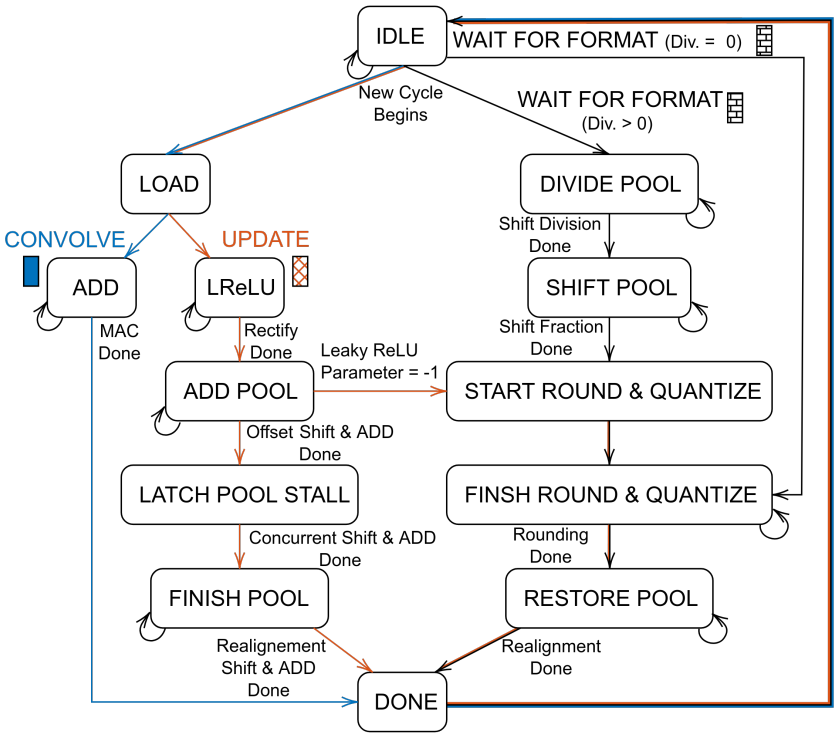



Figure 3.27: FSM control of the PE. Each colored path corresponds to a different control path depending on the state of the CNN control FSM.

During each system clock cycle, the processing element modifies its objective depending on the current state of the CNN control hardware as shown in Figure 3.27. During the `CONVOLVE` CNN state, the blue path is taken, where the processing element word-serially multiplies and accumulates activations in the accumulation register within the `ADD` MAC state. During the `Update` CNN state, the partial sum is added to the pooling register of the current active layer. The pooling accumulation and LReLU occur simultaneously by first rectifying the accumulation register in the `LReLU` MAC state, then shifting and adding this value to the pooling register. If the LReLU parameter is greater than 0, the pooling register of channels with negative accumulation values are stalled during shifting, such that the accumulator value is effectively divided by the parameter's set number of bits within the `LATCH POOL STALL` and `FINISH POOL` states. During the `Wait For Format` CNN state, the pooling register is normalized by bit shifting (`DIVIDE POOL`, `SHIFT POOL`), rounded (`START ROUND & QUANTIZE`, `FINISH ROUND & QUANTIZE`), and shifted back into place (`RESTORE POOL`). During the `WAIT FOR FORMAT` CNN state, a 9 bit partition of the pooling register is written to the feature shift register for export.

## 3.6 Data Interface

In some cases, the peripherals of a design can be as much work as the core of the system. Multiple clock domain transfers, complex command interpretation, and high data through put requirements, all made this block the biggest headache of the entire design. Unfortunately, this data interface became much more complex than necessary, wasting much effort and time. Although it more than meets the design requirements for the neural system, it became a limiting factor in being able to fully characterize the device at high data throughput rates. As a result, the maximum throughput of the core computational architecture was limited by this system, and not the core itself. As a word of advice: If your IO is causing you a huge headache, start over simpler.. If there is ever a design where you think you need to create your own bus interface/IO architecture, please consider using existing standards like AXI or AHB. With that word of caution out of the way, the system as it is designed is described in the following section.

### Data Interface Design

To facilitate data transfer into and out of the system at high enough speeds suitable for online validation of the architecture with a human in the loop, a custom flow-controlled data interface was constructed. This interface is also charged with decoding commands to properly interpret data packets. The interface commands are listed in Table 3.1 and concern both configuration and operation modes of the system.

Table 3.1: Data Interface Control Commands/States

| Command | Code (3b) | Description |
|---|---|---|
| Load Weights | 0x2 | Configure weight SRAM |
| Run FENet | 0x4 | Enter running state of system |
| Soft Reset | 0x5 | Reset core without resetting configurations |
| Configure FENet | 0x6 | Update model parameters within configuration registers |

The schematic of data interface is depicted in Figure 3.28. The flow control incorporates receive and transmission stall signals so that the system can adapt to a wide variety of transmission and system clock rates. The system needs to cross between the interface and system clock domain to retrieve feature data, as well as write configuration registers. This is accomplished with asynchronous queues which field the data, while state machines on the receiving end react to the presence of data on their queues.

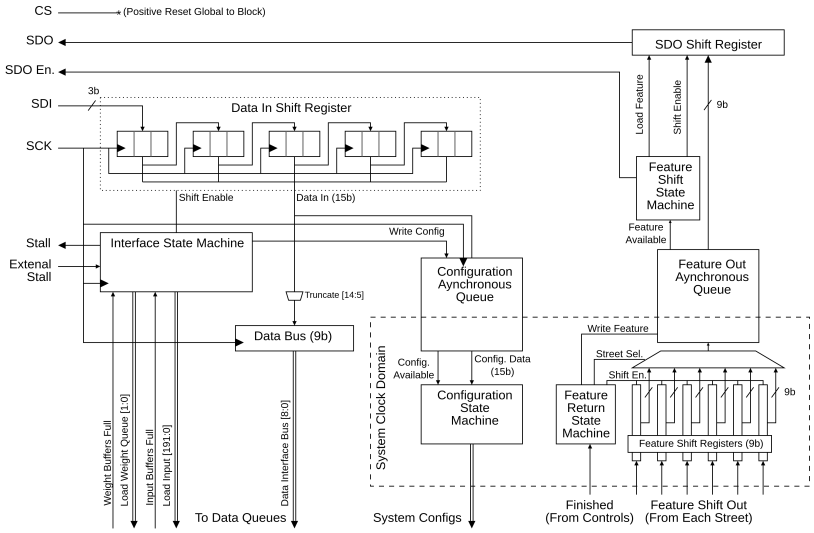



Figure 3.28: Schematic depicting the various components of the validation data interface. The ports on the top left are IO ports on the ASIC, while the ports on the bottom interface with the ASIC system.

The physical schematic of Figure 3.28 shows all the components of the data interface starting at the top left with the input shift register which de-serializes the SDI words. Below and to the left the shift register is the state machine which directs the inputs received by the interface to their respective destinations based on the interface command. If the data is destined for input channels or weight memory, it is latched to the data bus and the corresponding load input signal is pulsed so that the data is loaded into the correct channel queue. A configuration command directs the interface to load the input value into the configuration asynchronous queues which synchronize them with the system clock domain and writes the corresponding configuration data into the configuration registers. The configuration code decoding and write steps are directed by the configuration state machine located in the bottom-center of the schematic. The transmit components of the data interface are located on the right side of the schematic. Each street has its own feature shift out port. These features are de-serialized using a shift register, and then loaded into the feature out asynchronous queue to synchronize their values to the interface clock domain. The values in this queue are consumed by the feature shift state machine, which loads the values into the SDO shift register and shifted out of the ASIC.

There are 3 serial data in pins, and 1 serial data out pin. These pins follow a similar scheme to SPI where the output is driven on the falling edge of SCK, while the input is clocked in on the rising edge of SDK. This is to allow large skew ranges for the IO drivers. A chip select (CS) pin acts as an active high reset for the serial interface system. When CS is pulled low, the next 15 bits are interpreted as the data packet header with the first 3 bits corresponding the code located in Table 3.1. Following this code contains a data-field specific to the configuration command. and shown in Table 3.2. These commands allow the configuration of the FENet model, creating high flexibility in the FENet model shape and size which the architecture can implement.

Serial data is shifted in 3 bit increments through the SDI ports. Two stall signals control the flow of data depending on both the availability of data from the validation interface and the state of asynchronous queues in the receptive data destinations. These flow control signals were instantiated to allow for fine adjustment in the system and processing element clocks, while maintaining the same data flow rate of neural data.

Table 3.2: Configuration Command Bit Fields

| Name | ConFigure Code (3b) | Description |
|---|---|---|
| Kernel Width | 0x0 | Kernel width of each layer (starting at layer 0) |
| Stride | 0x1 | Stride for each layer |
| Divide | 0x2 | Divide normalization parameter for each layer |
| RELU | 0x3 | LReLU $\alpha$ parameter for each layer |
| Num. Cycles | 0x4 | Bin size in number of first layer strides |
| Num. Layers | 0x5 | Number of layers enabled (-1) |
| Channel En | 0x6 | One-hot encoded enable vector for channels |
| MAC Delay | 0x7 | Number of synchronization cycles for MAC sequence |

### Data Interface Performance and Limitations

Since this system is not intended only for validation purposes, its power domain is kept separate from the system. The validation system delivers the input data serially at a limited effective bandwidth of approximately 2.25 Mb/MHz as a result of dead time in the protocol caused by data caching cycles and stall checks, which can constrain the maximum achievable feature rate for models with high computational loads and high channel counts. However, this parasitic limitation is a function of the validation system interface, and not the processor architecture. The IO skew was measured to be a maximum 30 nS. With de-skew hardware implemented in the FPGA, the maximum clock frequency achieved with this validation interface was 42 MHz, maxing out the validation interface's bandwidth at 94.5 Mb/S which is 1.82× the maximum necessary rate for neural data (30 kSps × 9b × 192 channels). The power consumption of this data interface was measured at 0.9 V to be $6.14\mu\,W$/MHz. Furthermore, the validation interface is designed with the assumption that it will always operate at a frequency faster than the system clock. With this in mind, the hardware dataflow used to export the features does not have feature export flow control, resulting in the requirement that the validation interface must be faster ($\approx 9\times$) than the system clock to properly export features from all streets. Reducing the number of streets relaxes this requirement, but puts constraints on the operating frequency ranges of the interface clock, and therefore the power performance.

*Chapter 4*

# ALGORITHM OPTIMIZATION AND VALIDATION

This chapter presents the necessary explorations to trim the fat of the FENet model designed in [22] while maintaining performance. First, we take a look at how different configurations of the FENet model (kernel size, stride, number of layers) affect the minimum number of multiplications required for each feature set. In Section 4.2, FENet models are trained using automated design space exploration to identify new models which have lower complexity requirements, with similar performance. Section 4.3 compares the performance of the optimized FENet models to other commonly used feature extraction techniques, then the ASIC system is fully verified online with human subjects in the loop. To further reduce power, the FENet model's tolerance to lower neural data sampling rates, and the affect of gating neural channels on decoder performance. Finally, we explore how various model parameters affect the performance of the hardware architecture.

## 4.1 Complexity Analysis

This section presents the analytical methods to determine the algorithmic complexity of a given FENet model. This analysis is used to optimize models for improved power and latency performance. The following sections will first mathematically define the number of operations and resources necessary for the FENet workload based on model parameters. This is followed by simulation of the latency incurred from the padding operation.

**Operation Complexity**

The neural decoding environment requires high power efficiency and precision. FENet demonstrates superior feature extraction robustness since it does not depend on spike detection which becomes unreliable at low SNRs, yet feature generation from broadband neural data does not benefit from the sparsity of event-based processing. To mitigate this trade-off, the complexity of the FENet architecture is scrutinized to minimize power and hardware costs. While [22] mentions rough estimations for the number of MAC operations for a given FENet model, precise derivations for the number of non-zero-padding (NZP) MACs were not derived. These equations allow for the construction of a more accurate model the power consumption of a given model on this architecture. Therefore, it was necessary to derive these equations as are shown below. The majority of computational power of FENet is spent during the convolution which requires numerous multiply-accumulation (MAC) operations and significant memory access. The number of convolution operations for a given layer is shown in (4.1):

$$
\begin{aligned}
C_{spl} &= \left\lfloor \frac{min(K_l - 1,\ B_l)}{S_l} \right\rfloor \\
B'_l &= max(B_l - S_l C_{spl}\ ,\ 0) \\
B^r_l &= max(K_l - S_l + (B'_l) mod S_l\ ,\ 0) \\
C_{ssl} &= \left\lfloor \frac{B'_l}{S_l} \right\rfloor \\
C_{fpl} &= \left\lceil \frac{B^r_l}{S_l} \right\rceil \quad C'_{fpl} = \left\lfloor \frac{B^r_l}{S_l} \right\rfloor \\
B_{L+1} &= C_{spl} + C_{ssl} + C_{fpl}
\end{aligned}
\tag{4.1}
$$

where $B_l$ and $B_{L+1}$ is the input and output bin size for a given layer, respectively.

The $C_{spl}$, $C_{ssl}$, and $C_{fpl}$ terms define the number of convolution cycles contributed by the start-padding, steady-state, and finish-padding phases of the CNN for a given layer '$_l$'. Similarly, the number of MACs that those convolutions evoke are given in (4.2):

$$\begin{aligned} M_{spl} &= S_l C_{sp}(C_{sp}+1) \\ M_{ssl} &= 2 * K_l C_{ss} \\ M_{fpl} &= 2 * C_{fp}(B_l^r mod S_l) + S_l C'_{fp}(C'_{fp}+1) \\ M_l &= M_{spl} + M_{ssl} + M_{fpl} \end{aligned} \tag{4.2}$$

where $M_{spl}$, $M_{ssl}$, and $M_{fpl}$ define the MAC operations attributed to the start-padding, steady-state, and finished-padding phases of the computation for each layer.

For a given input bin length $B_0$, increasing the stride of the lowest order layers has an exponential effect on reducing the complexity of the model as it reduces the number of activations to higher order layers. Kernel size, on the other hand, does not have as dramatic an effect on the number of MACs but does significantly increase the size of padding since the number of padding activations is aggregated across layers.

**Padding and Latency**

Since most of the convolution computations occur in between the arrival of neural data, the latency between the arrival of the last sample of neural data within a bin and the completed feature is entirely dependent of the model's number of conclusional padding cycles. The number of MACs are well defined based on the model parameters using equations (4.1) and (4.2). Since different kernel length and stride combinations lead to the conclusional padding sequences to begin at different stages convolution, there does not exist a closed relationship between the number of non-zero-padding macs and model shapes, and the number of cycles required for conclusional padding. To define this behavior more precisely, a cycle-accurate simulator of the FENet architecture operating without IO limitation from the validation interface was build in Python. A Monte-Carlo simulation was run with various combinations of model parameters, and an estimator was constructed and tested for its prediction capabilities. The parameters were limited such that the kernel is at least as large as the stride to avoid edge cases.

Using the Monte-Carlo simulations, a closed form estimator was devised to help analytically determine the number of latency cycles. The hypothesized estimator uses the difference between the number of non-zero-padding multiply-accumulate operations including, and excluding the conclusional padding cycles. The number of NZP-MACs without conclusional padding is defined in equations (4.3):

$$\begin{aligned}
C_{spl_{nfp}} &= \left\lfloor \frac{min(K_l - 1,\ B_{l_{nfp}})}{S_l} \right\rfloor \\
B'_{l_{nfp}} &= max(B_{l_{nfp}} - S_l C_{spl_{nfp}}\ ,\ 0) \\
B^r_{l_{nfp}} &= max(K_l - S_l + (B'_{l_{nfp}}) mod S_l\ ,\ 0) \\
C_{ssl_{nfp}} &= \left\lfloor \frac{B'_{l_{nfp}}}{S_l} \right\rfloor \\
B_{L+1_{nfp}} &= C_{spl_{nfp}} + C_{ssl_{nfp}} \\
M_{spl_{nfp}} &= S_l C_{sp_{nfp}} (C_{sp_{nfp}} + 1) \\
M_{ssl_{nfp}} &= 2 * K_l C_{ss_{nfp}} \\
M_{l_{nfp}} &= M_{spl_{nfp}} + M_{ssl_{nfp}}
\end{aligned} \tag{4.3}$$

where the variables carry the same meaning those in equations (4.1) and (4.2), but do not include the finish padding components, and so carry the '$_{nfp}$' subscript. Using the NZP-MAC computations from (4.2) and (4.3), we can construct an estimator for the number of MACs that occur after the last input is received using equation (4.4):

$$Latency\ Cycles \approx \gamma \times \sum_{l=0}^{L} \frac{M_l - M_{l_{nfp}}}{2} \tag{4.4}$$

where $\gamma$ is a scalar value which compensates for loading, and formatting overhead cycles. Using the Monte-Carlo simulations, 30000 models were generated.

We found an $\gamma$ value of 1.136 estimates the latency cycles with an average error of 39.9%. Unfortunately, simulation remains the best estimator of latency for this architecture. The simulation, however, revealed the same $\alpha$ coefficient when using the actual number of MAC operations which occurred after the last input than those calculated in (4.3), indicating that $\gamma$ is a good estimator for padding overhead due to overhead caused by loading cycles in the control sequence.

## 4.2 Model Reduction and Retraining

This section details the methods and conclusions of parameter optimization of the FENet model. This section begins by introducing the automated hyperparameter optimization using the wandb framework with several choice models demonstrating a breadth of models and their use cases. Explorations into the number of necessary channels for decoding are shown, and their effect on decoding accuracy is reported. This is followed by a generalized analysis of how different model parameters affect latency and accuracy.

### Model Retraining

To explore the design space of FENet models, a parameter sweep was performed across all configurable hyperparameters of the FENet model using the wandb training framework with Bayesian optimization. These explorations resulted in significant reductions in computational complexity were selected and evaluated based on feature $R^2$ and expected power usage. Each model was trained on a 10-day dataset using 7-fold cross-validation. The selected models, shown in Table 4.1, were chosen to span a range of qualities that are useful for different feature extraction scenarios.

Table 4.1: Selected hardware-optimized models.

| **Model** | [22] | FENet-240 | FENet-66 | FENet-15 |
|---|---|---|---|---|
| $K_0$ | 40 | 40 | 36 | 10 |
| $S_0$ | 2 | 2 | 2 | 3 |
| LReLU Leak Slope | -1 | -1 | -1 | $-1/2$ |
| $K_1$ | 40 | 40 | 14 | 5 |
| $S_1$ | 2 | 2 | 2 | 3 |
| LReLU Leak Slope | -1 | -1 | -1 | $-1/64$ |
| $K_2$ | 40 | 40 | 16 | |
| $S_2$ | 2 | 2 | 2 | - |
| LReLU Leak Slope | -1 | -1 | -1 | |
| $K_3$ | 40 | 40 | | |
| $S_3$ | 2 | 2 | - | - |
| LReLU Leak Slope | -1 | -1 | | |
| $K_4$ | 40 | 40 | | |
| $S_4$ | 2 | 2 | - | - |
| LReLU Leak Slope | -1 | -1 | | |
| $K_5$ | 40 | 40 | | |
| $S_5$ | 2 | 2 | - | - |
| LReLU Leak Slope | -1 | -1 | | |
| $K_6$ | 40 | | | |
| $S_6$ | 2 | - | - | - |
| LReLU Leak Slope | -1 | | | |
| Total Weights | 560 | 480 | 132 | 30 |
| Total MACs | 30240 | 27120 | 9136 | 1250 |
| NZP-MACs | 19560 | 17960 | 7520 | 1176 |
| Pooling OPs | 417 | 379 | 210 | 91 |
| SRAM Writes | 528 | 489 | 327 | 222 |
| PE Clock Multiplier | - | 21 | 21 | 26 |
| Cycle Count/Feature | - | 11908 | 5449 | 1636 |

Operation costs are calculated assuming a bin size of 150 samples, corresponding to a 30 ms window at a 5 kSps sampling rate. Non-Padding-MACs (NP-MACs) refers to the number of multiply-accumulate operations that do not include the multiplications by zeros as a result of padding. Total MACs include padding and are performed in software implementations. Each pooling operation includes the LReLU activation, rounding, quantization, and accumulation into the pooling register.

Convolution padding is important for mitigating aliasing artifacts that result from binning neural data into fixed-length chunks. However, padding introduces additional multiply operations that affect the latency of computation. For FENet-66 and a bin size of 150, zero-multiply padding operations account for 12.6% of the total MACs. These operations are avoided altogether by trimming the convolution kernel size at the beginning and end of the feature computation. Each model was selected to represent different optimization goals:

**FENet-240** shows the highest overall decoding performance, but has a high model complexity.

**FENet-15** is the most efficient in terms of complexity and power, and still performs better than non-CNN-based feature extraction methods.

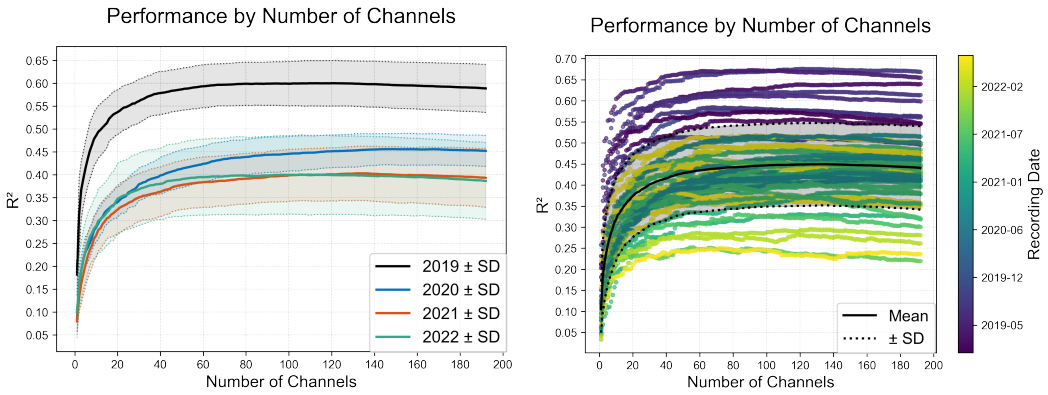
**FENet-66** balances decoding performance and computational cost.

These three models demonstrate the flexibility of the architecture to support a range of power-performance trade-offs depending on system requirements. **FENet-66** is chosen for its power efficiency while maintaining the majority of the feature extraction capabilities of larger models.

### Channel Gating

Neural activity related to kinematic decoding is spatiotemporally distributed within the motor cortex of primates [45] meaning that channels with information for a given decoding task are sometimes highly local, and remaining channels can be redundant. Additionally, each electrode may either be non-functional due to probe degeneration, functional, but uncorrelated with the desired decoding task, or functional, and relevant to the decoding task. The spatial maps of relevant channels are learned during the decoder training process, and are subject to change in medium-term time scales akin to hours and days due to the gradual immune response and micro-movements of the electrodes themselves. To take advantage of the sparse channel relevancy of neural recordings, the proposed architecture is granularly power-switchable to flexibly reduce power consumption of channels that are not highly informative.

We analyzed the performance of the kinematic decoding system as a function of the number of channels used after sorting each channel for its single channel decoding performance. The results of this analysis are shown in Figures 4.1 A and B. In the particular use case of kinematic decoding of thumb movement, decoder performance plateaus quickly at $\approx$ 64 channels. This highlights the benefit of being able to selectively power down channels within a decoding pipeline, allowing power to be spent selectively on channels that are only useful to ascertain a specific neural state.




(A) (B)

Figure 4.1: Decoding performance from limiting the number of channels with hardware implemented FENet-66 features:
(A) Average performance of each year based on a sweep of top channels. (B) Scatter plot of each day as the top channels are enabled sequentially. Color indicates the date the session.

### Hyper-Parameter Exploration

The FENet architecture differs from conventional CNN accelerators by processing time-series neural data as it arrives, without requiring large activation caches. As a result, 3 distinct timing metrics impact system performance: the time to process incoming data streams, the latency incurred during the padding phases of convolution, and the time required for the validation system to deliver input data to the asynchronous queues. In this work, those timing metrics combine to affect the feature rate defined as the speed at which the system can generate complete feature sets from streamed input data at a given set of clock frequencies. Feature rate depends on both data availability and the computational latency of the processor.

Table 4.2: Latency and minimum system clock frequencies required to achieve 33 FPS feature generation.

| **Model** | | FENet-240 | FENet-66 | FENet-15 |
|---|---|---|---|---|
| Padding Clock Cycles | | 7038 | 1135 | 111 |
| 30 kSps | System Clock [MHz] | 1.60 | 0.980 | 0.310 |
| | Latency (ms) | 4.4 | 1.2 | 0.4 |
| 10 kSps | System Clock [MHz] | 0.664 | 0.345 | 0.105 |
| | Latency (ms) | 10.6 | 3.3 | 1.1 |
| 5 kSps | System Clock [MHz] | 0.556 | 0.188 | 0.054 |
| | Latency (ms) | 12.6 | 6.0 | 2.1 |

Accurately modeling the performance, and clock rate requirements for a given set of model parameters, and neural data rates allows closed-loop optimization of FENet model design and hardware. To this end, a cycle-accurate model simulator was built to estimate the number of cycles required to extract each feature set, and how many of those cycles are spent on conclusional padding.

Intrinsic processing latency in the ASIC is dominated by the final padding phase, when no new input data is available for the current bin. Data that is not streamed during padding must either be externally cached or discarded, however, this issue would be mitigated through modification of the control FSM, to allow caching into the SRAM of the first layer during the conclusional padding phase. Because new data cannot be streamed during padding, optimizing the number of MAC operations during this phase is critical.

The number of clock cycles required to process the padding phase for each model is summarized in Table 4.2 along with the padding latency incurred while operating at the minimum frequency to achieve a feature rate of 33 FPS. Since all other computations are completed in-between the arrival of data-samples, the number of clock cycles of latency is constant and determined by the number of padding cycles for a given FENet model, regardless of the number of channels.

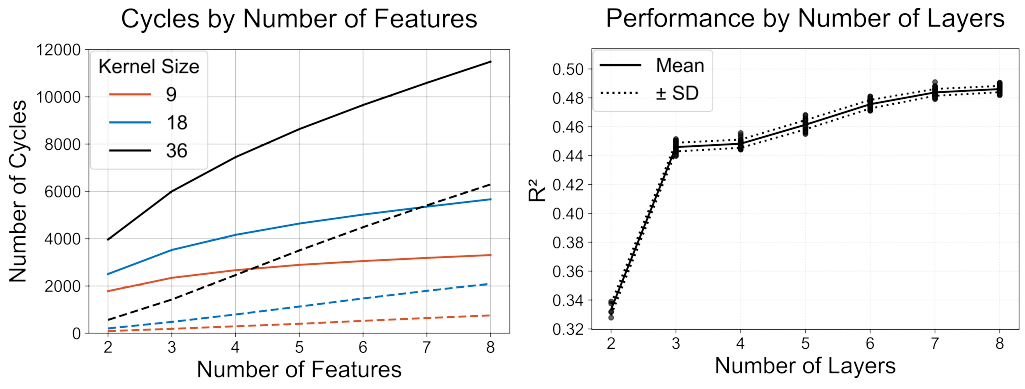
FENet-66 requires 6.2× fewer padding cycles than FENet-240, enabling feature generation at 33 FPS while operating the system clock at only 188 kHz with 5 kSps neural data and a padding latency of 6 ms. In contrast, FENet-240 requires a 2.9× higher system clock to meet the same feature rate due to its larger model complexity. These improvements in padding efficiency directly translate to lower operating frequencies and reduced dynamic power.

In Figure 4.2 A, we show the tradeoffs between the number of features, the total number of cycles required for each feature (solid line) and the number of those cycles that are necessary for padding (dashed lines). The minimum operating frequency of the system is related to the cycle count by equation (4.5):

$$f_{sys} = N_{features} * N_{cycles} \tag{4.5}$$

where $f_{sys}$ is the minimum operating frequency, $N_{features}$ is the desired number of features per second, and $N_{cycles}$ is the minimum number of cycles required of the model. Using this frequency, we can roughly determine the power and latency tradeoffs for a given feature rate.

While the effect of kernel size on accuracy is highly non-linear, the number of layers can in some degree be related to decoder accuracy. We explore this effect in Figure 4.2 B. We trained 51 models and held the kernel width and stride constant at 40 and 2, respectively, to tease out only the effect of the number of layers. We notice that there is a quasi-logarithmic effect on the number of layers to decoding accuracy, which reflects that the majority of neural information is captured in the lowest feature layers, with diminishing, but extant returns on performance as the number of layers is increased.




(A) (B)

Figure 4.2: Model Parameter Exploration:
(A) Cycle count for models with various hyperparameters. Solid lines denote the total cycle count, dashed lines indicate padding cycles. Kernel sizes are constant for all layers. Bin size: 150. (B) Effect of the number of feature layers on decoding performance with a constant kernel size and stride of 40 and 2, respectively. $R^2$ from 10 days of training data only.

## 4.3 Model Validation

Validation of this architecture both offline, and with patients in loop with the feature extraction architecture ensures that incorporating the architecture into a larger BMI SOC will perform with the expected benefits of the FENet algorithm. Two forms of validation are conducted with this system: offline validation, where a large data set of recorded neural data is processed with the ASIC to be tested for decoding performance, and online validation, where neural data from a patient is directly streamed from an FDA approved neural data collection system (Cerebus) into the ASIC, where features can be processed in real time. This validation allows verification of the system within closed-loop human control, where confounding factors such as extraction latency and quantization have the potential to disrupt the quality of kinematic extraction.

### Offline Validation

Offline validation allows large datasets to be tested with the hardware without the constraints inherent to online testing with humans in the loop. This allows a wide range of voltages and frequencies to be tested with the system to find the optimal operation points, as well as determine the sensitivity of the system to various sources of error like quantization, and error caused by undervolting various components. Furthermore, it allows comparison with standard feature extraction methods implemented in hardware so that the benefits of this hardware system can be assessed.
It is important to note that unlike spike detection and spike sorting methods, broadband feature extraction does not detect and classify spikes, but captures aggregate neural activity within broadband data, which makes direct comparison with these methods inapplicable. Therefore, to evaluate performance, we followed the cross-validated linear decoding methodology established in [22]. Features were generated from neural recordings and used to train and test a linear decoder. Decoder performance is measured by the coefficient of determination ($R^2$), described in (4.6):

$$R^2_{vx|vy} = \left( \frac{\sum (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum (y_i - \bar{y})^2 \sum (\hat{y}_i - \bar{\hat{y}})^2}} \right)^2 \tag{4.6}$$

which quantifies the correlation between the decoded and intended target velocities. Given the 2 degrees of freedom in center-out tasks (X and Y velocity), $R^2$ values for each dimension are combined into a single score via the root mean square, as shown in (4.7).

$$R^2 = \frac{1}{\sqrt{2}} \sqrt{(R^2_{vx})^2 + (R^2_{vy})^2} \tag{4.7}$$

Offline open-loop decoding analysis validates the ASIC feature performance across a large set of prerecorded data. Data from 48 center-out sessions [22] were used for benchmarking. Raw neural signals were first preprocessed by removing the first 2 Prinicpal Component Analysis (PCA) components of each array (for PCA-based CAR), followed by an 8th-order elliptical high-pass filter (80 Hz cutoff, 0.01 dB passband ripple, 40 dB stopband attenuation) and batch normalization.

Hardware-generated features were reduced from $N$ to one feature per channel using a PLSR model. To mitigate overfitting, a single averaged PLSR model was trained offline on one session, then applied across all 48 sessions. A linear least-mean-squares regression decoder was trained with 10-fold cross-validation for each session to compensate for non-stationary effects of the implant due to micro-movements.

**Chronic Stability and Feature Comparison**

Feature performance was compared against established feature extraction methods in the literature including wavelet transform broadband feature extraction, spiking-band power[3, 8], threshold crossings, and multi-unit activity. Wavelet transform features were generated by loading our ASIC with the hardware-friendly Haar wavelet transform which has 3 layers with kernel size 2, totaling 4 output features. We use the same number of layers as FENet-66 to keep the decoding dimensions the same. We also used the same sampling rate as the target rate of FENet-66 (5 kSps) such that only the effects of using trained kernels are compared. Spiking-band power features were generated by first filtering the neural data at 1 kHz, and downsampling to 2 kSps, then averaging the magnitude of neural recordings within the 30 mS time bin. Threshold crossing features were generated by counting crossings over an adaptive threshold set at $-3.5\times$ the root mean square of the neural signal in 30 mS bins. Multi-unit activity features were generated by spike sorting the threshold crossing events utilizing the sorting and clustering algorithms used in [64].

Chronic performance stability is illustrated in Figure 4.3. Notably, threshold crossing performance degrades sharply by year three after implant (session 35), coinciding with the loss of separable single-unit activity (SUA) on the 2 arrays. FENet-66 consistently maintains a higher average decoding performance after the loss of single-unit activity in the fourth year (sessions 35-48) of 0.404 compared to wavelet transform (0.370), spiking-band power (0.315), multi-unit activity (0.134), and threshold crossings (0.083).

The Normalized Performance Retention (NPR) show in (4.8):

$$NPR = \frac{R^2_{i^{th}\ year}}{R^2_{first\ year}} \tag{4.8}$$

provides a measure of the stability in performance for each feature extraction method. Comparing the first-year average $R^2$ (FENet-66: 0.605, wavelet transform:0.578, spiking-band power:0.552, multi-unit activity: 0.552, threshold crossings: 0.509) to the fourth, the NPR for FNet-66 is 0.66, whereas the other methods have a NPR of 0.64, 0.57, 0.48, and 0.16, respectively. This highlights the benefit of the FENet hardware in maintaining decoder stability over long implant lifetimes.

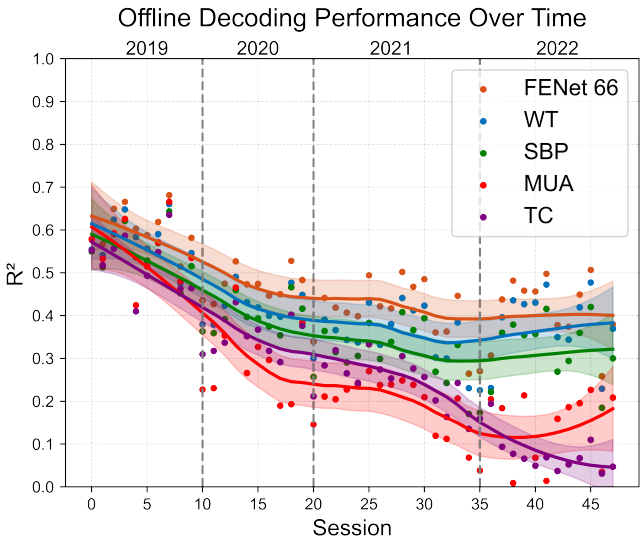



Figure 4.3: Cross-validated decoder $R^2$ performance over four years post-implantation. Locally Estimated Scatterplot Smoothing (LOESS) fits and confidence intervals are shown for each feature type.

### Sample Rate Reduction

FENet software was previously tested at 30 kSps [22], consistent with other techniques in the literature [41, 69, 25, 21]. This rate was originally used because it is the maximum sampling rate provided by the FDA-approved Blackrock Cerebus system. To test robustness to lower sampling rates, data sampled at 30 kSps were reduced by integer factors while adjusting bin sizes to maintain an output feature rate of 33 Features-Per-Second (FPS). Downsampling was performed by first filtering the data by $\frac{1}{2}$ the downsampled rate with a low pass anti-aliasing filter followed by decimation.

In Figure 4.4, we explore the effect of sampling rate on the average decoding performance over 48 neural recording sessions spanning 4 years. We observed high stability in performance for FENet 66 and 240 versus sampling rate down to 5 kSps. This is a result of the fact that a typical neural spike sampled at 30 kSps has a waveform that occupies approximately 40 samples, which is similar to the kernel width of the first layer of FENet 66 and 240. Their ability to accentuate real neural spikes from noise is related to the kernel's similarity to the average neural spiking shape. Furthermore, models with more layers are able to maintain their feature extraction ability at lower sampling frequencies because the power of neural spiking shapes is redistributed to higher-order layers.

FENet 15, has first layer kernel size of 10, which at 30 kSps is unable to fit an entire waveform into a single convolution; this explains why it performs worse at 30, kSps, and better when the size of a typical neural spike after downsampling is similar in length to the first kernel. However, since it has the least number of layers of all the models, less power is able to be redistributed to higher order layers when the sampling rate is reduced further from its optimal value.

We compare the robustness of models FENet-240, FENet-66, and FENet-15 to lower sampling rates in Figure 4.4. For comparison, the session-averaged $R^2$ performance of wavelet transform, spiking-band power, multi-unit activity, and threshold crossing features are shown as stars for reference as shown in Figure 4.4.

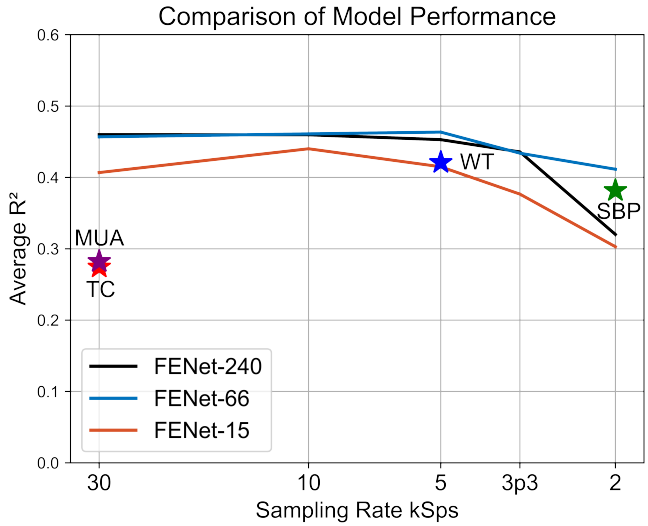



Figure 4.4: Cross-validated decoder $R^2$ performance of FENet models versus sampling rate. The average $R^2$ performance of other features from Figure 4.3 is shown as starred points for reference.

At 5 kSps, FENet-66 achieves an average $R^2$ of 0.46, maintaining 98.7% of the cross-validated offline performance of FENet-240 while requiring 2.6× fewer MAC operations. FENet-66 also outperforms wavelet transform (10%), spiking-band power (18%), threshold crossings (38%) and multi-unit activity (41%), achieving a total average $R^2$ of 0.382, 0.282, and 0.275, respectively. To fairly compare spiking-band power to FENet, we also measured the performance of FENet-66 on 2 kSps data and found the average $R^2$ to be 0.411, which is still 8% better than spiking-band power. This performance is attained while only consuming 346 $\mu$W over all 192 channels (1.8$\mu$W per channel). FENet-15, although lower in decoding performance, still outperforms all other hardware implemented methods at 5 kSps, while also minimizing power consumption to 219 $\mu$W (1.14 $\mu$W per channel). The 4 layer Haar wavelet transform does remarkably well for its simplicity, achieving 90% the performance of our trained kernels, while requiring 177 $\mu$W (0.92 $\mu$W per channel).

**Online Validation**

Offline validation allows analysis of a broad set of system parameters, whereas real-time closed-loop analysis ensures generalization of offline results within a real-world setting that has a number of confounding variables such as latency between feature generation and kinematic prediction. We tested the feature extraction system by decoding kinematic intent using ASIC-generated features and returning visual feedback to the patient by updating the position of a cursor on screen within a center-out task.

Our participant (JJ) was implanted with two 96-channel MEA Utah devices in their motor and peripheral parietal cortices six years before this online evaluation. All procedures were approved by Caltech's Institutional Review Board (IR20-0983). An initial decoder was trained in an open-loop trial where the participant (JJ) imagined tracking an on-screen cursor with his thumb without feedback. ASIC-extracted features from this trial were used to train a linear decoder. In subsequent closed-loop trials, the cursor position was updated in real time based on decoded kinematics. The decoder was fine-tuned through successive trials, gradually reducing assistance, ultimately achieving fully autonomous control.

The online setup is depicted in Figure 4.5. The neural data is collected with the Blackrock Neural Systems amplifier which initially digitizes the electrical signals measured from the implanted array with integrated first order Butterworth filters with band cutoff frequencies of 0.3 Hz and 7.5 kHz. These digital signals are then streamed via an optical link at 30 kSps to the Cerebus neural signal processor, which further filters the digital signals with a second order Butterworth filter with cutoff frequencies of 250 Hz and 5 kHz. These streams were then broadcast onto a local Ethernet network where they are received by the validation server. The validation server preprocessed the streams by common average referencing them, and forwards the referenced neural streams to the FENet ASIC for feature extraction. The extracted features are subsequently returned to the validation server for broadcasting onto the Ethernet network where they are received by a separate computer running the decoding experiment. Data were sent to the FPGA-based validation server where CAR was applied before being forwarded to the ASIC for feature extraction. Features were transmitted back to the trial computer for real-time decoding and cursor control.

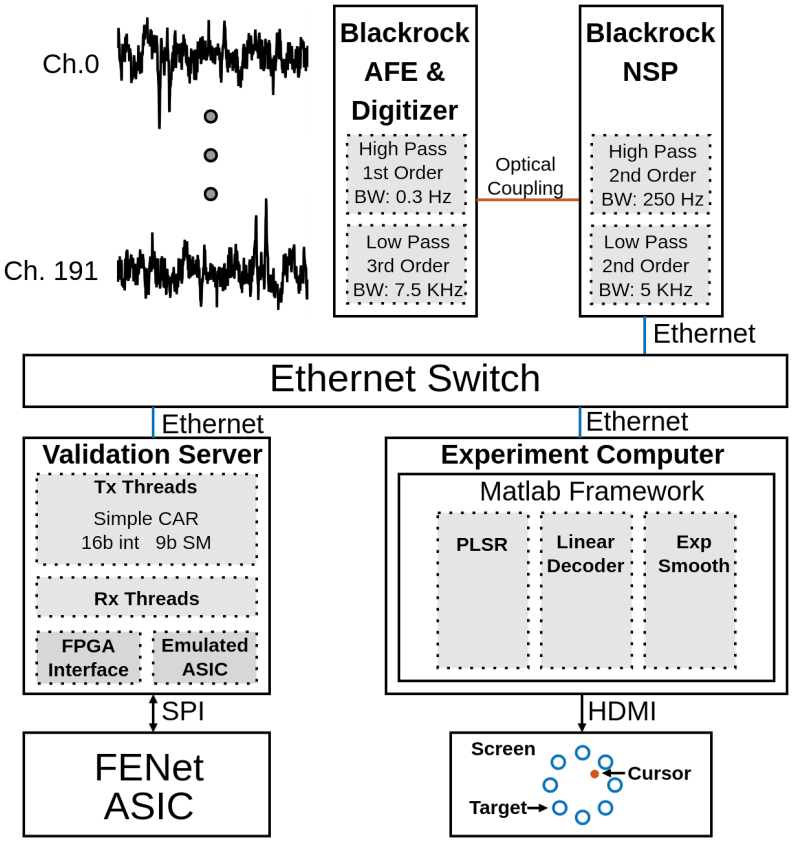



Figure 4.5: Data flow for neural data retrieved during an online session.

The decoding pipeline was implemented in Matlab on the experiment computer. Features were dimensionally reduced using PLSR, then decoded to kinematics with a linear decoder, exponentially smoothed [37], and displayed as cursor position to the patient.

The linear decoder was first trained using an open-loop trial and had a combined x-y $R^2$ performance of 0.71. The neural data for this open-loop trial was later processed using the software-FENet implementation of [22], which yielded a cross-validated $R^2$ of 0.70. This shows that the hardware implementation maintains open-loop decoding performance similar to software-bound implementations even six years after implant. Spike analysis of this neural data yielded no detectable single neurons and a total of 119 non-separable spike channels with a mean and median SNR of 1.12 and 0.94, respectively. There were only two channels with an SNR greater than 3 (maximum 5.25). The ability of our hardware to generate usable features for kinematic decoding from such noisy signals exemplifies the importance of stable decoding hardware for implantable devices. Utilizing the spike activity filtered from noise, we performed open-loop decoding which yielded a cross-validated multi-unit activity $R^2$ performance of 0.43.

We show the closed loop kinematic decoding trial utilizing FENet-66 processing 10 kSps neural data streams in Figure 4.6. The FENet ASIC generated 33 FPS for all 192 channels. The mean time-to-target is measured at 1.00 seconds, with an $R^2$ of 0.66. All 192 channels were used in decoding without gain normalization on the validation server.

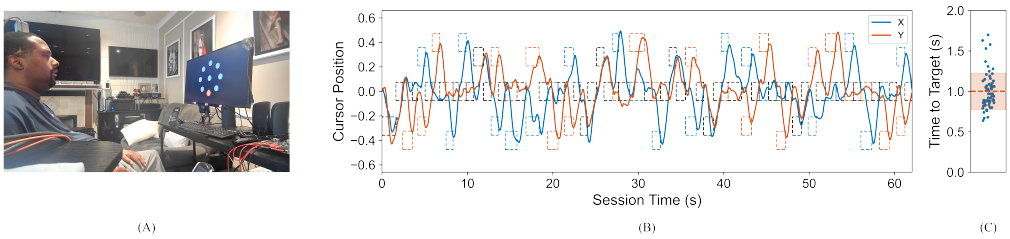


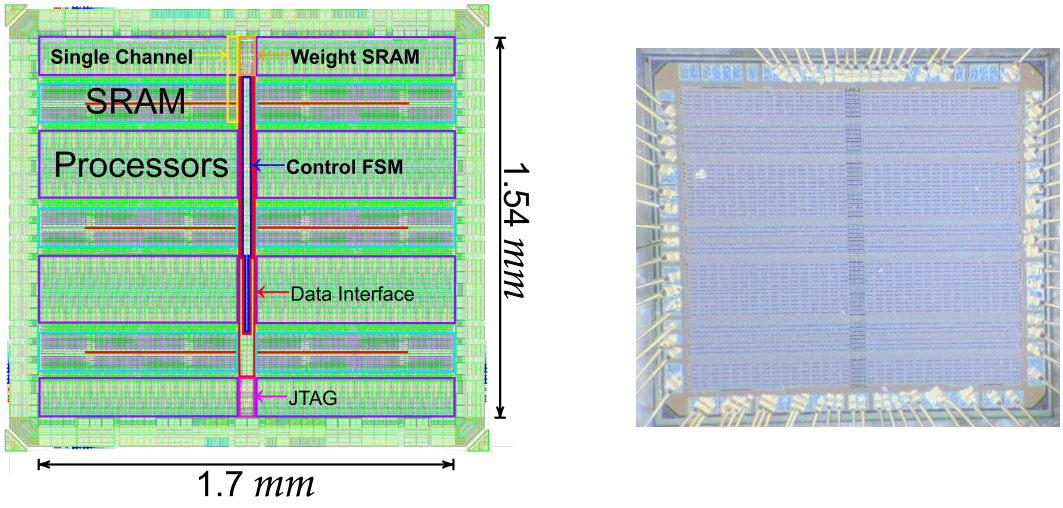

Figure 4.6: Performance of online decoding session completing a center-out kinematic control task:(A) Research participant controlling a cursor utilizing ASIC for kinematic decoding in a center out task. (B) Online closed-loop decoding session using FENet ASIC in loop. Boxes represent the target where the height is the size of the target in its represented dimension, and the width represents the time it took to reach the target; color corresponds to the x and y dimension of the cursor control. (C) Time-to-target plot for all 63 targets with a mean time-to-target of 1.00 (seconds).

*Chapter 5*

# HARDWARE MEASUREMENT AND ANALYSIS

This chapter presents the characterization of the architecture as well as measurements of the system in operation in offline and online neural decoding environments. It further shows results characterizing the difficulty of the neural decoding problem and the FENet workload. The successful implementation of FENet with low power and area was confirmed both offline and online. The measurement results show that our architecture is capable of implementing the FENet workload with minimum SRAM memory requirements at a reasonable power consumption, which contributes to its viability in the area constrained neural decoding pipeline.

The FENet ASIC was fabricated in 65 nm LP CMOS with the dye graph shown in Figure 5.1. The dye graph depicts the spatial allocation of hardware for the FENet ASIC. SRAM (highlighted in turquoise) is tightly packed with processors, and arranged in a ribbed fashion. A small portion of the ASIC is used for control hardware and the data interface, and is distributed along the spine of the ASIC.




(A) (B)

Figure 5.1: (Fabricated ASIC in 65 nm LP CMOS process:
(A) Dye graph of ASIC with various components labeled. The processing elements (purple) are tightly coupled with activation SRAM (light blue). The validation interface (red) broadcasts neural data to each street in between each activation SRAM. (B) Picture of ASIC fabricated in 65 *nm LP 9M* CMOS.

## 5.1 Hardware Validation Server

Online validation of brain-machine interface algorithms is vital to ensuring that processing metrics such as latency, quantization error, and data flow jitter do not significantly diminish closed-loop kinematic decoding. To enable the online validation of our feature extraction system, the fabricated chip was integrated into a Linux environment, capable of buffering, preprocessing, and routing neural data into the fabricated feature extractor. This system must also handle data retrieval and broadcasting of the processed features to the task computer dedicated to running the BMI experiment.

### System Overview

A schematic overview of the offline validation system is depicted in Figure 5.2 with a picture of the system in Figure 5.3. This system is built around an AMD Zynq UltraScale+ MPSOC ZCU106 evaluation board. The SOC-FPGA chip that this board supports, is charged with hosting the validation server software which communicates with the custom communication interface built to ferry data into and out of the ASIC.

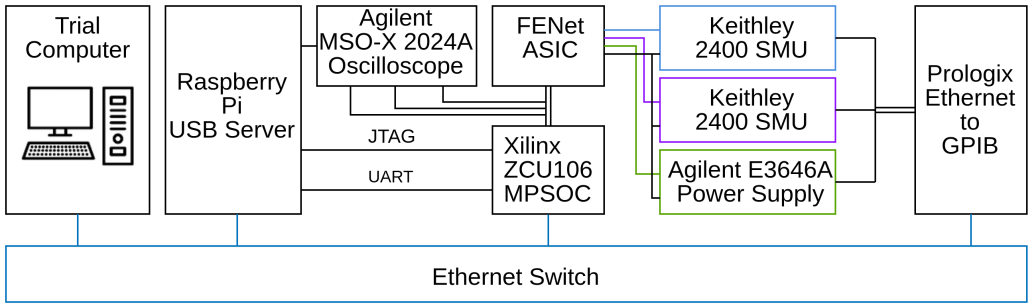



Figure 5.2: Simplified schematic diagram of offline validation system showing connections.

Supporting equipment used for power source and measurement is controlled remotely by the task computer via a local Ethernet connection. This is to facilitate numerous trials with different system configurations remotely. The memory and processing element core voltages are measured using two separate Keithley 2400 source measurement units (SMU), while the interface domain was powered using an Agilent 3464 power supply. The power supplies were connected to the local Ethernet network using a Prologix Ethernet-GPIB server. The UART and JTAG connections were facilitated through USB ports on the Zynq board.

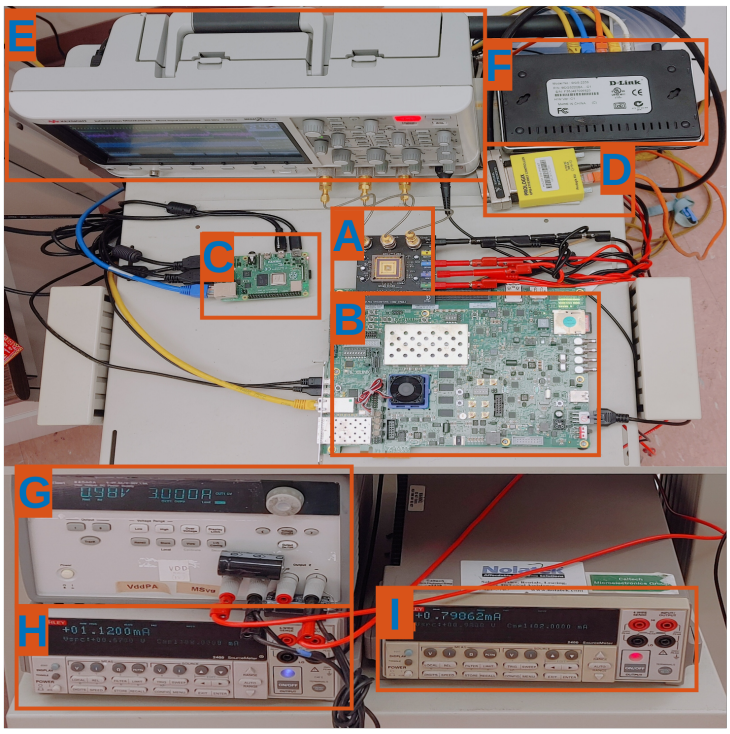



Figure 5.3: Offline validation setup with highlighted components. (A) FENet ASIC, (B) Validation Server, (C) USB-Server, (D) Ethernet-GPIB Adapter, (E) Oscilloscope, (F) Ethernet switch.

The USB port on the task computer USB port was discovered to sporadically de-power the power its USB ports when idle, which would brick the USB connection until the port was manually disconnected and reconnected. To resolve this issue, a Raspberry Pi 4 board is utilized as a USB server. An emulated replica of the ASIC logic was implemented on the FPGA to verify ASIC outputs in real time. The ASIC was powered with separate voltage domains for the validation interface, processing elements, and memory. Power measurements were collected after a 30-second stabilization period to ensure steady-state conditions. IO and validation system power were excluded from all reported values.

**System Requirements**

To be viable for online testing, this system must meet throughput and latency requirements to facilitate responsive online cursor control with system requirements listed in Table 5.1. In addition to the listed requirements, this system must integrate with the Cerebus FDA-Approved neural acquisition system provided by Blackrock Microsystems which is used to interface the patient with our experimental setup. This entails decoding the proprietary Ethernet packet structure of the Cerebus system and repackaging the contained data into a format compatible with the ASIC. Furthermore, it must allow for remote configuration of algorithm configuration parameters (weights, kernel size, stride, division coefficients, leaky ReLU slope, number of channels enabled), and system parameters like the frequency of each of the three clocks driving the system.

Table 5.1: System requirements for validation server

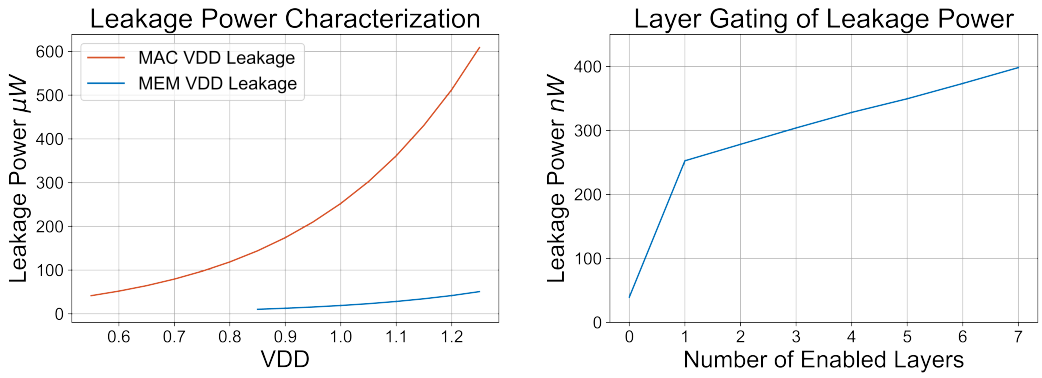
| | |
|---|---|
| Maximum Neural Packet Rate | 60000 Packets per Second |
| Maximum Neural Data Rate | 11.52 MB/s |
| Ethernet Protocol Support | UDP and TCP |

## 5.2 Static Power Characterization

Efficiency in operations per watt does not belie all the important power metrics necessary in determining if an architecture is optimal for a particular workload and operation environment. For instance, an architecture utilizing fully parallel arithmetic units can lower dynamic power since they require fewer clock cycles to complete the computation and can utilize low-power multiplier topologies like the modified Booth-Wallace tree. However, these parallelized systems require more gates and therefore more static power than their more compact serial counterparts. In the specific use case of the FENet workload where the number of multiplications per model is relatively lower than most CNNs, this static power becomes particularly wasteful as the multiplier spends most of its time idle between clock cycles.

**Power Gating**

The leakage power of this architecture with all channels and levels enabled is shown in Figure 5.4 A. Figure 5.4 B shows the leakage power with different layer configurations. The processing element (MAC VDD) and memory (MEM VDD) domains have typical operating voltage of 0.65 V and 0.9 V, respectively, when processing FENet workloads. The leakage power of each domain under these operating conditions is 64.4 $\mu W$ and 12.7 $\mu W$, respectively, totaling 77.1 $\mu W$ over the whole device. The total switching power of a functional system can range from 100-2000 $\mu W$ depending on the model and sampling rate of the FENet workload. Since highly power-optimized models can operate at the low end of that spectrum, leakage power can account for up to 40% the total power of the system. Fine-grained control of layer power distribution allows conservation of leakage power in smaller models while maintaining model flexibility. Channel-wise power gating further allows ultra-low-power workloads to operate with minimal waste by unused processing channels. As shown in Figure 4.1 Section 4.2, shortly after an array is implanted, the number of channels for high performance can be very low ($\approx$ 64 channels). For example, FENet-66 operating on top-64 channels with 5 kSps data on recordings can achieve 99% decoding performance while needing only 140 $\mu W$. Without channel-wise gating, this would increase by 40% to 197 $\mu W$.




(A) (B)

Figure 5.4: Leakage power characterization: (A) Leakage power with all 192 channels and 7 layers enabled. (B) Leakage power per channel with MAC and MEM VDD at 0.65 V and 0.9V, respectively. The $0^{th}$ layer index indicates the leakage power of a disabled channel.

## 5.3 Dynamic Power Characterization

This section characterizes the chip architecture under various operating conditions and work loads. This allows a deeper understanding of the power consumption of various components and provides insight into which components could benefit from optimization in future iterations of the design. As a result of the highly granular power switching capability of the system, it is possible to directly determine the power consumption with similar granularity.

### Power Characterization Methodology

To accurately measure the dynamic power, the measurements from the prior section measuring the static power are first subtracted from any power measurements. Since our dynamic power measurements use voltage and channel enable sweeps to improve accuracy, a voltage-dependent model was created to accurately predict the leakage current to remove. FENet-66 is used for all dynamic power measurements, so the leakage current model has 3 layers enabled.

Taking advantage of the highly configurable nature of this architecture, the various components of power consumption can be finely ascertained to build a dynamic power model which can predict power consumption based on a set of configurations and model parameters. This is accomplished by first setting an operating condition (frequency, model, packet rate, VDD, ect.), then modulating the number of channels and observing the change in power. Doing so with the FENet ASIC architecture results in a curve as seen in Figure 5.5.

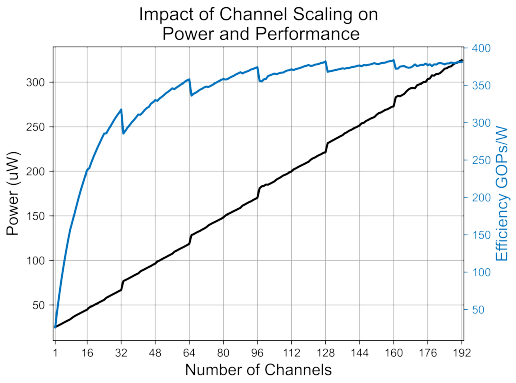



Figure 5.5: Power and efficiency scaling by the number of channels enabled. Number of channels is scaled linearly, such that each street is filled sequentially.
Operating conditions: Model: FENet-66; Sampling Rate: 5 kSps; Clock Frequency: Sys. 0.188 $MHz$, MAC 3.948 $MHz$, Intf. 6 $MHz$; VDD: MEM 0.9 V, MAC 0.63 V

As observed, the power and efficiency scaling is nearly linear, with small point-wise nonlinearities at intervals of 8, and larger point wise nonlinearities at intervals of 16. This corresponds to blocks becoming enabled in the case of the former, and entire streets (along with the street clock buffers), becoming enabled in the later. These non nonlinearities allow for the dissection of the power consumption of each of the components under various work loads. We can model this behavior using the equation in (5.1):

$$System\ Power\ (\mu W) = \alpha + \kappa * N_{Channels} + \beta * N_{Blocks} + \sigma * N_{Streets} \tag{5.1}$$

where $\alpha$ is the baseline power of the system and $\kappa$, $\beta$, and $\sigma$ are the scaling factors for the number of channels, blocks, and streets enabled, respectively. These parameters are all functions of the operating conditions mentioned earlier. Teasing out the relation of these parameters to different operating conditions defines the characteristic power behavior of this architecture, allowing for deep analysis on the system, and which components and methods are ripe for optimization.

The $\alpha$ parameter corresponds to the baseline power of the system, which includes the leakage power of the entire chip and the always-on components of the system which include the weight SRAM, control finite state machines and scheduler. It further includes the clocking architecture which distributes clock signals to each of the gated clock buffers at the beginning of each street. The $\kappa$ parameter corresponds to the power that can be gated with granularity, which includes the FENet module, its leakage power, the asynchronous queue, and the switching activity of the SRAM channel that is modulated. The $\beta$ parameter defines the switching activity for each block's control signals, while the $\kappa$ parameter primarily corresponds to the switching power required to drive each street's clock and control signals.

To extract parameters, the number of channels is swept up to 64, allowing for 64 measurements of the delta from increasing channels, 8 from increasing blocks, and 2 from increasing streets. These parameter extractions are accomplished with the following steps:

**Step 1 ($\kappa$):** Find the difference in power between each channel as they are enabled so that the difference in power comes only from this additional channel (Figure 5.6 A). Filter the differences that require enabling additional block or streets (indexes are multiples of 8). Find the mean of differences, removing outliers if necessary to find a best fit parameter.

**Step 2 ($\beta$):** Project the power of the system by number of channels using the parameter found in Step 1 (Figure 5.6 B). Subtract this projection from the total power. The remaining function will be a stepped function, where each step corresponds to the increase in power from blocks and streets. Subtract each of these levels from each other, filtering the levels which correspond to an increase in streets. Find the average difference as before.

**Step 3 ($\sigma$):** Project the power of the system by the number of channels and blocks using the parameters from Steps 1 and 2 (Figure 5.6 C). Again, subtract this projection from the power. The remaining step function has a single step that corresponds only to the power difference as a result of the increase in the number of streets.

**Step 4 ($\alpha$):** Finally, project the power of the system by number of channels, blocks and streets from the above steps (Figure 5.6 D). The remainder corresponds to the baseline power of the system.

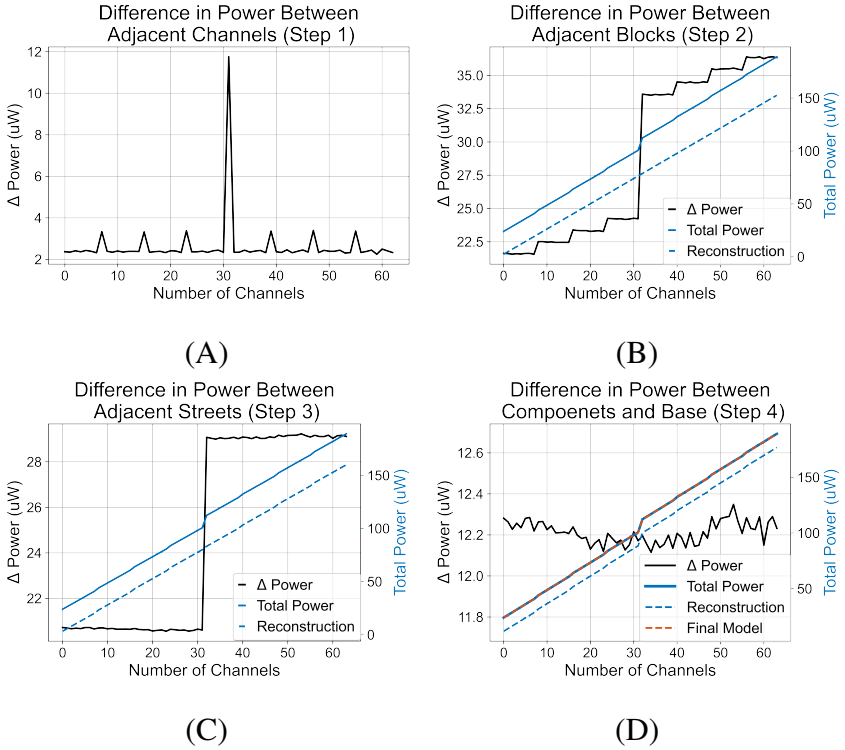



Figure 5.6: Power parameter extraction procedure:
Operating conditions: VDD: MEM 0.95 $V$, MAC 0.70 $V$, Frequency: System 0.182 $MHz$, MAC 4.37 $MHz$, Interface 24 $MHz$ Packet Rate: 33.7 kSps (unbound) (A) (Step 1) Difference between adjacent channels in the MEM power. (B) (Step 2) Difference between adjacent blocks in MEM power. Reconstruction of power model using only the $\kappa$ parameter (dashed blue line). Actual MEM power is the solid blue line. (C) (Step 3) Difference between adjacent streets in MEM power. Reconstruction of power model using both the $\kappa$ and $\beta$ parameters (dashed blue line). Actual MEM power is the solid blue line. (D) (Step 4) Difference between model using only components, and the actual power. Reconstruction of power model using the $\kappa$, $\beta$ and $\sigma$ parameters (dashed blue line). Actual MEM power is the solid blue line. Final model is shown as the dashed orange line.

### Clock Distribution Power

Clock distribution is the baseline for dynamic power characterization. Measuring clock distribution power provides an important baseline to compare against when the workload is introduced. Table 5.2 provides $\alpha$, $\kappa$, $\beta$, and $\sigma$ components for each clock measured at 1 V, separated by MEM and MAC VDD domains. Since the clock networks of pooling layers are also configurable, the layer-dependent component for affected parameters is indicated with the subscript '$_L$'.

Table 5.2: Component parameter extraction of clock distribution power.

| Clock Name | MEM Power ($\frac{\mu W}{MHz V^2}$) | | | | | MAC Power ($\frac{\mu W}{MHz V^2}$) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha_{mem}$ | $\alpha_{Lmem}$ | $\kappa_{mem}$ | $\beta_{mem}$ | $\sigma_{mem}$ | $\alpha_{mac}$ | $\kappa_{mac}$ | $\kappa_{Lmac}$ | $\beta_{mac}$ | $\sigma_{mac}$ |
| System | 3.242 | -0.274 | 0.077 | 0.765 | | | 0.011 | 0.005 | | |
| MAC | 0.314 | | | | 0.081 | | 0.097 | 0.005 | 0.501 | |
| Interface | 0.125 | | 0.106 | | 0.337 | | | | | |

As an example the channel power parameter $\kappa_{MAC}$ for the processing element domain is calculated with the equation (5.2):

$$\kappa_{MAC} = \kappa_{mac} + \kappa_{Lmac} \times (N_L - 1) \tag{5.2}$$

where $N_L$ is the number of enabled layers (which is one less than the number of output features).

To verify the model accuracy, the error was measured for the 5847 measurements taken to derive the clock power model. Figure 5.7 depicts the probability distribution of error for this model that shows good agreement with the measured values with a slight bias towards underestimation of power, but in total the model estimates the power to within 10% error.

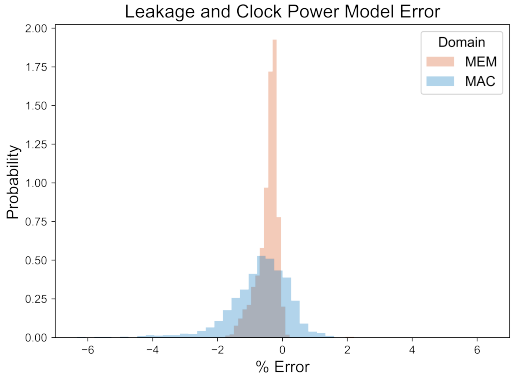



Figure 5.7: Probability distribution function of clock power error for both the MEM and MAC power domains.

One interesting observation is the negative relationship between the number of layers enabled and the baseline power consumed. This particular quirk in the design was ultimately traced back to gating logic which controls the clock of the padding state machine. Disabled layers of this state machine were left to continuously update with the default value, evidently wasting 274 $\frac{nW}{MHzV^2}$ of power per layer. While this is an unimportant fact of the architecture, it highlights the sensitivity of this method of analysis to interrogate which architectural components can be improved upon.
More importantly, this analysis belies an unusually high power consumption by the distribution of the validation interface while no data is present. Further investigation shows that each clock gate within an asynchronous queue is preceded by a clock buffer. Given that the validation interface only interacts with each channel once every load cycle, a system with all channels enabled will ultimately require 191 clock network cycles more than necessary. Thankfully, if this system were fully integrated into a neural decoding pipeline, parallel data sources would only need to interact with the asynchronous queues at the rate of data arrival, removing these erroneous cycles entirely. However, it is worth noting that the true power of the architecture is augmented by an amount estimated in equation (5.3):

$$MEM_{P_{waste}} = \frac{N_{Ch.} - 1}{N_{Ch.}} \times (\kappa_{MEM} \times N_{Ch.} + \sigma_{MEM} \times N_{St.}) \times F_{Intf.} \times V_{MEM}^2 \quad (5.3)$$

Where $N_{Ch.}$ is the number of channels, $\kappa_{MEM}$ and $\sigma_{MEM}$ are the memory domain channel and street parameters, respectively, $F_{Intf.}$ is the interface clock frequency in megahertz, and $V_{MEM}$ is the memory domain voltage.

**Power Efficiency**

The following subsections analyze the system power under computational load. The breakdown of power for the components of each channel is shown in Figure 5.8 A and is estimated by post-place and route simulation at 1 V and 1.2 V and was observed to be relatively consistent across voltages within each respective domain. The total power ratio was measured directly with a MEM and MAC voltage of 0.9 V and 0.63 V, respectively.

The decoder was trained using the FPGA features and validated with those generated by the undervolted ASIC. For this performance, the feature extraction ASIC consumed 346.2 $\mu$W to generate features for all 192 channels at an efficiency of 335 GOPS/W where a MAC operation is two OPS with a padding latency of 6 mS. The processing element consumes 179 $\mu W$ (52%) of the total power. These power results exclude the power contributions of the validation interface and IO.

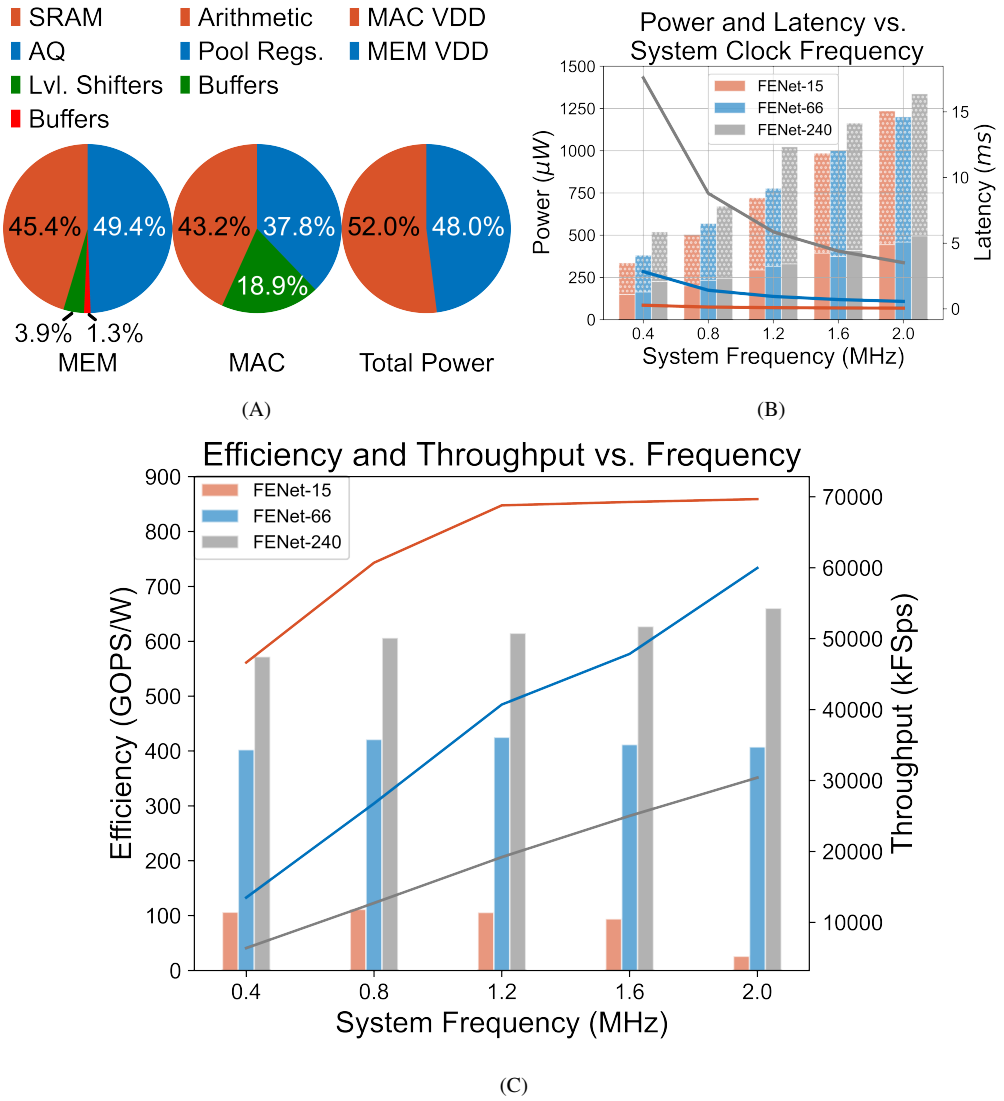



Figure 5.8: Power and efficiency characterization:
Operating conditions: Model: FENet-66; Sampling Rate: 5 kSps; Clock Frequency: Sys. 0.188 $MHz$, MAC 3.948 $MHz$, Intf. 6 $MHz$; VDD: MEM 0.9 V, MAC 0.63 V (A) Power breakdown of a single channel separated by VDD domain. (B) System power and latency at different operating frequencies. MAC voltage is scaled so $R^2$ maintains 98.8% performance. Bar graph denotes the power with top and bottom bars denoting MAC and MEM domains, respectively. (C) Efficiency in GOPS/W plotted as the bar graph and throughput plotted as lines in kilo Feature Sets per Second (kFSps) of various models.

The power consumed in various device configurations is shown in Table 5.3. Power was not measured directly during the online test, but the power draw for the same number of channels and sampling rate was measured to be 586 uW (3.05 $\mu$W per channel). In a highly optimized case, where only top-64 most informative channels are used streaming neural data at a 5 kSps sampling rate, the feature extraction power was reduced to 140 $\mu$W (2.2 $\mu$W per channel), while maintaining an average $R^2$ of 0.354 over all 48 sessions. Alternatively, all 192 channels can be operated at 2 kSps with 178 uW (0.93 $\mu$W per channel), with an average $R^2$ of 0.41. Reducing the device to 1 channel shows a minimal operating power of 25.2 $\mu W$, demonstrating the operation floor of this device.

Table 5.3: Power of various configurations using FENet-66.

| Channel Count | S. Rate (kSps) | Freq. (MHz) System//MAC | Voltage MEM//MAC | Total ($\mu$W) | Ch. ($\mu$W) |
|---|---|---|---|---|---|
| 192 | 2 | 0.095//2.0 | 0.90//0.63 | 178 | 0.93 |
| 64 | 5 | 0.188//3.984 | 0.90//0.63 | 140.0 | 2.2 |
| 192 | 5 | 0.188//3.984 | 0.90//0.63 | 346.2 | 1.8 |
| 192 | 10 | 0.345//7.25 | 0.93//0.65 | 586.4 | 3.05 |
| 192 | 30 | 0.980//20.58 | 0.98//0.67 | 1584.0 | 8.25 |

Gray highlighting denotes sampling rate used in online testing.

The FENet ASIC running the FENet-66 model substantially cuts the data rate necessary for transmission by 37.5× when operating on 5 kSps neural streams and 225× when operating on 30 kSps neural streams. Features are successfully generated at a rate of 33 FPS, which is sufficient for rapid, fine motor control.

We explore the power requirements of different models at various system clock frequencies in Figure 5.8 B. The MAC clock was maintained at a multiple of the system, consistent with the PE clock multiplier listed in Table 4.1. The data interface frequency was maintained at $9 - 14\times$ the system frequency. We further limited the neural data packet rate to 5 kSps, regardless of the maximum throughput of the ASIC, so that our system generates 33 FPS over the entire range of operating frequencies to match realistic data rates. MAC VDD was scaled so that $R^2$ performance is maintained above 98%. Since the interface speed was limited, MEM VDD was maintained at 0.9V across all frequencies.

We also measured the performance of the system using each model over the same system frequency range, without constraining the neural data rate. In this case, we optimized the interface clock frequency to the minimum frequency at which the processor can remain computationally limited, maximizing efficiency. The throughput and efficiency at various system frequencies are shown in Figure 5.8 C. FENet-15, FENet-66, and FENet-240 were measured to each require a minimum energy of 24 $nJ$, 42 $nJ$, and 82 $nJ$, respectively, for each feature set per channel. Noting the values in Table 4.1, we calculate the max efficiency to be 104 $GOPS/W$, 424 $GOPS/W$, and 661 $GOPS/W$, respectively (1 MAC = 2 OPs). This range in efficiencies correlates to the ratio of MAC operations, to the total number of cycles each feature set requires.

**Voltage Sensitivity**

Two core voltage levels are used to minimize processing power (MAC VDD) while maintaining the voltage headroom necessary for low leakage SRAM (MEM VDD). The voltage sensitivity of the ASIC is depicted in Figure 5.9 A. The measured sensitivity is consistent for system clock frequencies less than 700 kHz, which is fast enough to operate FENet-66 with a padding latency of only 1.62 ms. No errors were observed for MAC VDDs greater than 0.65V within this operating range. The mean squared error (MSE) between the ASIC and FPGA-reference generated features are plotted alongside the cross-validated decoder $R^2$ using data from the first session of the offline data also used in Figure 4.3. The decoder features show robustness to total MSE values less than 100 such that the $R^2$ performance maintains 98.8% of its value. As such, the minimum voltage values for the MAC and MEM VDDs are chosen to be 0.63 V and 0.90 V, respectively.

To better understand the voltage-delay relationship of this architecture, the maximum operating frequency was determined for a range of operating processing element voltages as shown in Figure 5.9 B. The maximum operating frequency was determined by sweeping the voltage until no discrepancies are observed between the features produced by the embedded architectural duplicate and the ASIC. This data shows a nearly linear relationship between the VDD and the switching speed. Since the construction of the validation system is designed for the relatively slow operating rates of neural signal processing, the measurement of this ASIC is IO limited and the processing element clock is limited to 100 MHz.

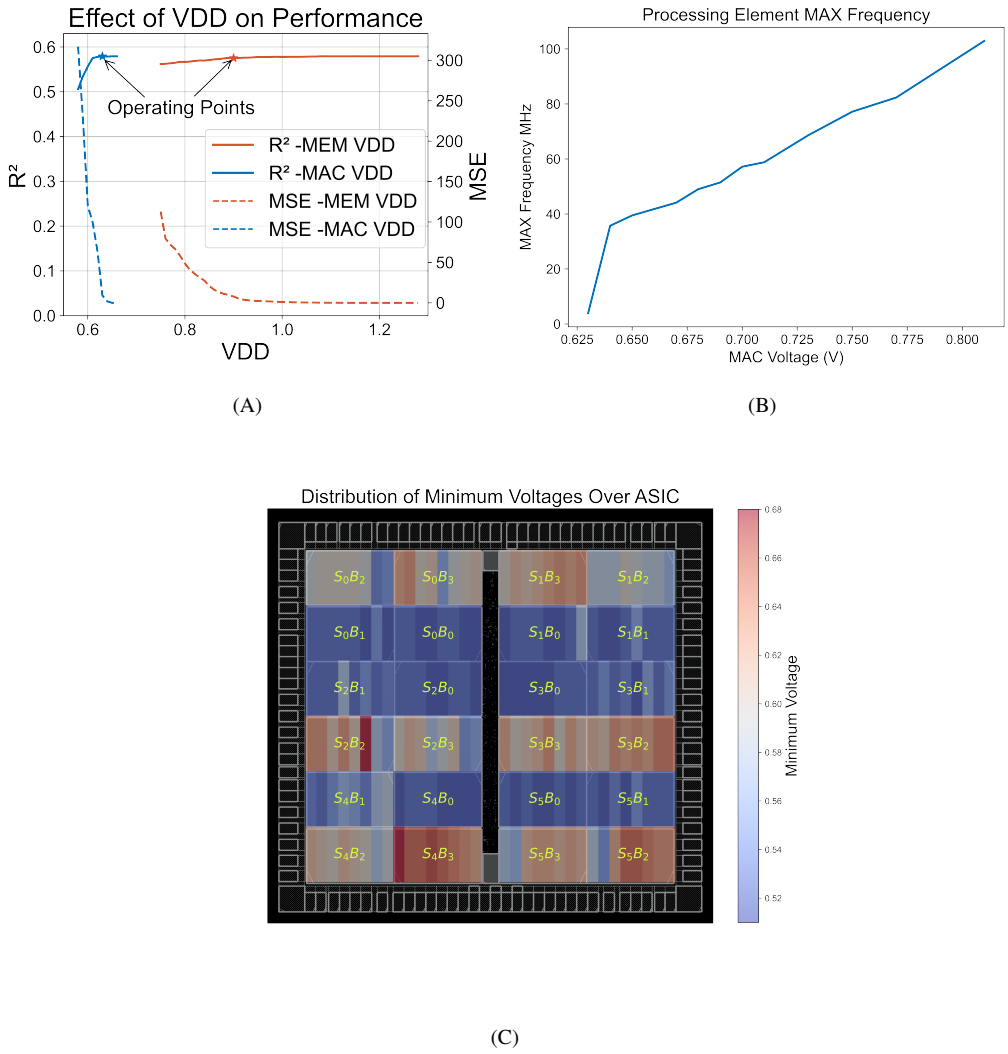



Figure 5.9: Voltage-Performance Characterization:
(A) VDD scaling effect on feature quality. Solid lines denote $R^2$, while dashed lines denote MSE of the features. First session of offline data is used for analysis on effect. (B) Maximum processing element frequency scaling with voltage. Memory VDD is fixed at 1.2 V. (C) Spatial distribution of minimum operating voltages across one ASIC sample.

The memory voltage was also swept with respect to system clock and interface clock frequency and showed no affect on VDD within the operating frequencies supported by the maximum processing clock of 100 MHz, and the IO limitations of the interface clock frequency (system frequency < 4 MHz, interface frequency < 42 MHz). This is due to the error becoming dominated by the low leakage SRAM headroom requirements, and not because of the switching limitations of the relatively slow system clock and interface clocks.

Finally, to ascertain the effects of the data and control distribution networks, the spatial voltage sensitivity was mapped across the chip. In this experiment, an optimal clock configuration was used with the FENet-66 workload, which means the interface clock was set to the minimum speed which does not bottleneck computational throughput (system clock: 0.4 MHz, processing element clock: 8.4 MHz, interface clock: 12 MHz). The memory domain VDD was fixed at 1.2 V as the processing element VDD was swept. The minimum voltage is determined when the rate of mismatch of a channel to the embedded architectural duplicate becomes 0.

The spatial distribution of minimum voltages under these operating conditions is mapped in Figure 5.9 C. Although it is important to note that the specific mapping is sensitive to process variation, certain architectural trends can still be observed. One particular trend is the fact that minimum voltage drops dramatically by $\approx 0.1\ V$ at the boundary between blocks 1 and 2 of each street, denoted as $S_nB_1$ and $S_nB_2$, respectively. This is likely a result of the extra long routing required to make the bend between block 1 and 2. Upon reflection, the centralized controls did not have a real need to be in the center of the design. Instead, placing the controls on one of the two sides of the ASIC would alleviate this routing issue. Furthermore, these results are indicative of the limitations of chaining these blocks far apart across a fully integrated SOC. Special care should be taken to re-optimize the signal integrity of blocks that require extra inter-block routing.

The architecture was tested using only half the channels in each street, and showed successful computation of features with a decoding $R^2$ 98% the original value while operating with a minimum MAC VDD of 0.56 V and a MEM VDD of 0.85 with a 0.4 MHz system frequency consumes 197.4 $\mu W$. These low operating voltages increased the GOPS/W efficiency of the FENet-66 workload from 424 GOPS/W to 621 GOPS/W while generating 69.9 frames per second, allowing processing of features from all channels within the same amount of computation time. FENet-240 with 0.8 MHz system clock frequency can produce 64.5 features per second requires only 353.9 $\mu W$ to operate at these voltages effectively, pushing its efficiency from 661 GOPS/W to 949 GOPS/W. Adjusting the power consumption by removing the unnecessary interface clock power with equation (5.3), we get an estimated power of 294 $\mu W$, which would put the architectures theoretical maximum efficiency at 1141 GOPS/W if corrections were made to the interface clock and buffer scheme.

## 5.4 Implementation Analysis

This section analyzes the architectural efficacy in the context of its application in brain-machine interfaces. The area and scalability of the architecture are important factors to consider when determining the viability of an architecture intended for implantable neural signal processing. These factors determine how well an architecture is able to integrate into a complete processing system on a chip, as well as how easily the system can be modified to support the increasing number of neural recording channels in modern brain-machine interface systems. This section is concluded with a comparison of the FENet ASIC feature extraction hardware with the prior state-of-the-art.

### Area Costs

The ASIC supports up to 192 neural streaming channels and occupies a total core area of 2.62 $mm^2$. Each channel occupies 12801 $\mu$m of area with the processing element requiring 7165 $\mu m^2$ (1447 gates), and the 4 element asynchronous queue and level shifters further consume 1144 $\mu m^2$ (121 gates), which is 56% and 9% the total area per channel, respectively. The 256-element SRAM consumes 3790 $\mu m^2$ which is 30% the channel area, the remainder being used for buffering and signal gating. The control logic and weight SRAM occupy an area of 31626 $\mu m^2$ (2423 gates) and 100082 $\mu m^2$, respectively.

### Architecture Scaling

Our architecture completes all possible computations at the same rate as data arrival. Conclusional padding cycles, defined only by the model, delineates the latency and therefore remains constant when scaling the system. All controls for the system are centralized and broadcast to each channel, with each channel block designed to be self-contained. Adding additional streets requires only buffering the control signals from the central FSM. As a result, the area of our system scales linearly, with a projected 1024 channel system requiring an additional 5.59 $mm^2$. Furthermore, each channel has an individual neural data port. With our validation system limited by IO, our channel count is ultimately constrained by the design specifications of the data interface. Integrating this FENet MACRO into an SOC with independent data sources would allow each channel to accept neural streams in parallel.

**Power Scaling**

One major consideration in this design is the scalability of power efficiency. Power scaling is especially important in chronically implanted high-channel-count neural systems, since probes can degrade to the point they no longer provide useful information to the decoder. With the models derived in 5.3, we can extrapolate the power scaling of the system. Targeting the minimum energy point for implementing FENet-66 at 5 kSps, the projected feature extraction system with 1024 channels, we would simply increase the number of streets (32 channels and 4 blocks per street) from 6 to 32. This predicts our system power to be around 1.67 $mW$.

**Comparison With Other BMI ASICs**

There is currently a surge in the development of processors targeting low-power edge applications [34, 27, 63, 65]. Our feature extraction chip is a domain-specific architecture optimized to implement FENet in the neural decoding environment. To the best of our knowledge, this is the first system which integrates multi-level global average pooling accumulators and dual mode convolutional data paths for each channel. Our dual mode processing elements generate intermediate activations for higher order layers, while simultaneously computing output features. With an 8 layer output stationary processing element, we entirely avoid re-fetching intermediate activations to generate the output features. Through intensive hardware reuse, channel-level-power scaling granularity, and unique data streaming structure, we optimized this architecture for the neural decoding environment, where other conventional CNN processors can be too bulky or memory intensive for the FENet workload. We optimized the data flow for FENet feature generation to achieve high kinematic decoding accuracy and stability with long implant lifetimes. Table 5.4 compares the proposed FENet architecture with existing hardware implementations of neural feature extractors.

The FENet algorithm is trained on data from Utah arrays with large probe spacing (0.4 mm) and therefore does not see significant inter-channel correlations of spike signals from the same neuron [9], however, training of FENet with presence of inter-channel correlations is an interesting topic for future investigations. Spike detectors and spike sorters like those found in [68, 69, 57, 63] are able to distinctly identify neural sources, and firing patterns, which is well suited for systems with high inter-channel correlations, but require high SNR for accurate detection and sorting.

The calibration free SD system in [68] has a power and area cost of 6760 $\mu m^2$ and 0.038 $\mu W$ per channel, respectively. Their system employs adaptive thresholding techniques, which showed maintained detection accuracy for 200 days. However, chronically implanted neural probes used in our study have mean noise levels of 89% six years after implant, which is much higher than the 20% tested in [68]. Even with our most advanced adaptive thresholding techniques, FENet features outperform SD (TC features) by 487% after 4 years.

SPB calculated in [3] has remarkably low complexity in relation to its performance, achieving 3.68 $\mu W$ and a scaled area of 2370 $\mu m$ per channel. Spiking band power features further reduce each bin of neural data down to a single feature. For a 30 ms time bin sampled at 2 kSps, this reduces the dimensionality of the neural data 60×. The simplicity of this algorithm accentuates its utility when power and area constraints are high and decoding precision is less pertinent.

Other neural interface modalities employ signal compression such that low data rate representations of the neural data can be transmitted then reconstructed with little or no loss. The system in [58], utilizes an Autoencoder (AE) to compress local field potentials (LFP) at 0.076 $\mu W$ and 0.02 $\mu m^2$ per channel with a compression ratio of 19.2. The compression system is designed specifically for LFP as it relies on the spatially correlated nature of these signals. The system allows for lossy reconstruction of the original signal with a signal-to-noise distortion ratio of 15-19 dB.

While our system primarily focuses on feature extraction, the systems in [3, 51, 63], incorporate on-chip decoders to fully integrate the decoding pipeline. The system in [3] achieved an average cross validated correlation of 0.29-0.49 with 1D, and 2D kinematic decoding trials, respectively, using a steady state Kalman filter. In [51], a decoder using distinctive neural codes and a linear discriminant analysis classifier were able to achieve 31-class handwriting classifications at 1 classification a second with 91.3% accuracy. This system was able to achieve this while only consuming 0.44 $\mu W$ and 1500 $\mu m^2$ per channel.

Table 5.4: Comparison with other state-of-the-art neural feature extraction ICs.

| Metric | **This Work** | TBCAS22[3] | TBCAS19[57] | TBCAS24[58] | TBCAS22[63] | TBCAS23[68] | JNE25[69] |
|---|---|---|---|---|---|---|---|
| Process | 65 | 180 | 32 | 180 | 22 | 65 | 65 |
| Implementation | Digital ASIC | Digital ASIC | Digital Sim. | Digital Sim. | Digital ASIC | Digital Sim. | Digital Sim. |
| Number of Channels | 192 | 93 | 1 | 96 | 16 | 128 | 8 |
| Channel Area $\mu m^2$ | 12801 | $28443^{\alpha}$ | 2570000 | 20000 | 14000 | 6760 | 6450 |
| Scaled Area$^{\phi}$ $\mu m^2$ | 12801 | $2370^{\alpha}$ | 8481000 | 1667 | 92394 | 6760 | 6450 |
| Channel Power $\mu W$ | 1.8 | 3.68 | 2.78 | 0.076 | 2.79 | 0.038 | 0.532 |
| Resolution (bits) | 9 | 16 | 6 | 10 | 8 | 1 | 1 |
| Sampling rate (kHz) | 5-30 | 2 | 24 | 24 | 20 | 7 | 24 |
| Feature Type | FENet | SBP | MUA | LFP | MUA | TC | TC |
| Algorithm$^{\gamma}$ | CNN | MAV | SS OSort | AE | SS | SD NEO | SD TEO & SWT |
| Avg. Feature $R^2$ | 0.446 | 0.382 | 0.275 | - | 0.275 | 0.282 | 0.282 |
| Feature NPR | 0.66 | 0.57 | 0.48 | - | 0.48 | 0.16 | 0.16 |
| Feature Rate FPS | 33 | 20 | Async. | - | Async. | Async. | Async. |
| Bin Size (samples) | $150^{\beta}$ | 100 | 64 | - | 64 | 16 | 80 |
| Supply Voltage (V) | $0.63//0.9^{\beta}$ | 0.625 | 1.16 | 1.8 | 0.63 | 1.8 | 1.2 |
| Clock (MHz) | $0.188^{\delta}$ | $0.068^{\eta}$ | 0.024 | 0.004 | 0.400 | 0.896 | 0.200 |
| Latency (ms) | 6.0 | 0.5 | 1.3 | - | 0.07 | - | 0.05 |
| Validation Model | Human | Primate | Synthetic | Primate | Rat | Synthetic | Primate |

$^{\alpha}$Calculated from feature extraction hardware only.
$^{\beta}$Configured for sampling rate of 5 kSps with FENet-66.
$^{\delta}$System clock frequency. Mac frequency for FENet-66 is 21x the system clock frequency.
$^{\eta}$Spiking band power feature extraction unit runnning at 2.9 MHz.
$^{\phi}$Scaled to 65 nm process using methods in [56]
$^{\gamma}$Mean Abolute Value (MAV); Auto Encoder (AE); Nonlinear Energy Operator (NEO); Teager Energy Operator (TEO); Stationary Wavelet Transform (SWT)

*Chapter 6*

# CONCLUSION

Neural interfaces are a burgeoning frontier that provides opportunities to alleviate suffering caused by injury or illness to the brain or nervous system. As these systems become a reality and move away from purely an academic setting, the longevity of these systems becomes extremely important. This dissertation presents the development of a scalable, low power 1D CNN-based feature extraction ASIC optimized to process broadband neural data streams with low power and area overhead. The architecture memory and scheduling scheme is optimized to implement the FENet algorithm on hundreds of continuous streams of data, which reduced memory requirements by 5× over conventional architectures, while the processing element further reduces memory access by 2× through a unique dataflow construction tailor built for the FENet workload. Through retraining and optimizing the FENet algorithm, the complexity requirements of the model implemented by this hardware was reduced to require half the number of multiply accumulate operations, while still generating neural features with state-of-the-art stability even six years post-implantation, where the maximum SNR of the multi-electrode arrays over all 196 channels was 5.25, with a with a mean and median SNR of 1.12 and 0.94, respectively. We validated the hardware-optimized FENet models and the ASIC that implemented them online through closed-loop cursor control with a human subject.

The power consumption of the ASIC was 341.2 $\mu W$ (5kSps) to generate neural features at 33 FPS for all 192 channels at a latency of 6 ms, fast enough for accurate and responsive kinematic decoding. The feature extraction hardware is highly power-scalable, providing flexibility to the decoding system. While the system does not dynamically re-configure itself, the models, and channel counts could be updated by a central control system of a more complex SOC that integrates the FENet hardware into its neural decoding pipeline. Early in an implant's life-cycle, when only few informative channels are necessary, power can be optimized to achieve quality decoding performance with low drain on system power. Later in an implant's life-cycle, when noise is high, more channels can be enabled to maintain performance at a linear expense in power.

Brain-machine interfaces have evolved expeditiously over the past few years. This work provides one important block of the neural decoding pipeline that significantly reduces the bandwidth of downstream decoding components, while maintaining much of the important information found in broadband neural data.

## 6.1 Current State of the Project

This work is a contribution for a larger ambition project to construct a neural brain processing system on a chip, complete with analog front end, signal preprocessing, and kinematic decoding. The current state of the ASIC implementation of the decoding pipeline covers the FENet feature extraction. Additional developments should include the analog front end and recording, followed by digital filtering and common average referencing, with the signals generated by this hardware then ready to interface with the FENet processors built in this dissertation. Following feature extraction, the features should be dimensionally reduced with PLSR, which can be done with matrix multiplication built into the existing FENet hardware protocol.
To accomplish this, a modification to the processing element data path must be made to allow the features stored in the pooling registers to be accessed as inputs to the processing element. Additionally, the ability to subtract bias values should be incorporated into the processing data-path. Finally, a decoder can be integrated into the system. Since data is streamed, it is important that the FENet hardware is quickly made available to processing the next batch of neural activations. For this reason, I believe a decoder should be build into separate hardware that implements any variety of decoding algorithms, from a simple linear decoder, to another deep learning algorithm such as a transformer or recurrent neural network.

## 6.2 Future Directions

While FENet shows its promise for use in neural feature extraction, I believe there is much more fruitful research in the space of combining simplified wavelet decomposition (like that of an 8 layer Haar transform) with a recurrent neural network. This combination enjoys the data compressive benefits of multi-resolution analysis in the Haar layer with data-driven feature extraction in the recurrent layer. Furthermore, this architecture would seamlessly integrate with the computational specialties of spiking neural networks and event-based processing.

At the writing of this manuscript, direct-digitization front-ends are showing promise in their ability to provide resource-efficient data conversion. Therefore, it will be advantageous to design feature extraction and neural processing algorithms that are capable of seamlessly accepting and processing the time-based spike encoding schemes generated by these front ends.

## 6.3 System Improvements

The system developed in this dissertation sets the framework for more complex neural decoding systems. Not only is the feature extraction hardware ready for integration into such an SOC, but the testing infrastructure for fully validating the system with human subjects will provide a significant head start for future BMI endeavors. The FENet ASIC has several areas that have opportunity for optimization which could reduce power, improve latency, or mitigate the drawbacks of the final padding cycles on the flow of the system.

### SOC Integration

This architecture is designed for efficient implementation of the FENet dataflow at the edge of data collection. As such, its benefits are fully realized when integrated into a neural decoding system on a chip (SOC). This includes integration of analog front ends and signal preconditioning, as well as back end digital decoding. The FENet architecture benefits from this integration since it is able to accept independent data streams for each channel. Since the current FENet architecture is straddled with IO limitations, the ability to independently load data in parallel, reduces the interface clock rate by N times (where N is the number of channels). Since the interface clock distribution was measured to be up to 30% the total power, system integration would entirely reduce this glutenous source of power consumption.

Furthermore, SOC integration allows for advanced reduction in transmission bandwidth down to only the kinematic extraction values. As the field of neuromodulation continues to advance, complex edge-decoding techniques can be introduced into the decoding pipeline [46] to provide even better kinematic decoding performance. A more advanced system redesign can allow for the reuse of the FENet processing elements for these 2D workloads. The major modification of the system architecture would be in the data network. The addition of a configurable network between the MACs and SRAM would allow the processing elements to exploit reuse opportunities within the 2D workflow.

**Read-Write Pipelining**

The FSMs generating read and write addresses have one cycle delay before initiating a read or write command. Furthermore, each MAC requires 2 cycles after each SRAM read cycle to finish processing the last element, and apply LReLU, quantization, and rounding to the accumulation register. As a result, the SRAM is inactive for 15% of the FENet sequence.

Two approaches can help mitigate this inefficiency. The first approach would be to pipeline the address generation segments of the FSM to remove the dead time before write operations. The second approach is to pipeline addresses for the following convolution cycle. During the final two computational cycles, the data for the next computational cycle can be read from SRAM, and cached locally. During the write back cycles, the processing are then free to process the already cached data for the next sequence, increasing overall utilization of both memory and processing resources.

**SRAM optimization**

One key design consideration was the choice of SRAM used for activations. While the sizing of the SRAM is optimal for this dataflow, the choice of SRAM speed is also worth considering. Since the SRAM compiler was sourced from TSMC, there was no way to validate the SRAM with analog simulators since no analog models are provided. As such, the decision to use the low leakage SRAM was made based on the power of the SRAM within the specified operating range ($1.2\ V \pm 10\%$), as this range is the only range guaranteed by TSMC. However, our measurements show that the SRAM is fully capable of operating at-least 25% below its nominal voltage. For this reason, the use of the high-speed SRAM which has higher leakage but uses IO peripherals with lower threshold voltages and therefore larger headroom should be explored as to remove the need for dual VDD rails, and level shifters across the entire design as well as allowing the system voltage to be scaled potentially to the same voltage as the processing elements.

Moreover, the system was designed with the constraint that it must fit large models such as FENet 240, however, it is evident that much smaller models are capable of similar performance with much lower memory requirements. This memory space can be reclaimed to reduce area and power, or put to use for different aspects of the neural decoding pipeline.

**MAC Parallelism**

The RTL design of this system was made with the ability to instantiate multiple MACs at a time without substantially changing the system. This ability was originally designed to allow the processing of multiple layers at a time to improve performance. There were a few factors that introduced impetus to the full adaptation of this feature including the low overall processing requirements and the fact that dual-port SRAM requires higher power per access, and the fact that any intermediate caching scheme would put restrictions on the stride length. Ultimately, flexibility and compact area were chosen over power efficiency and performance due to a lack of certainty on what the most efficient FENet algorithm would ultimately become.

Sacrificing configurability in the stride length, an activation reuse mechanism could cache one stride of activations locally, and use those activations to process the next convolution in parallel with the first. Each layer of MAC parallelism would reduce the number of activations and latency by this factor. Since the abstract structure of the system is already setup for such parallelism, implementation of this improvement would take minimal effort and has the potential for the largest improvement in efficiency. Another potential instance of MAC parallelism is the potential to use unused SRAM space to store multiple channels of neural data.

**Minimum VDD Reduction**

While often used to save area, standard cell gates which support more than 2 inputs increases the headroom of the subsequent processor. Removing these gates all together will push the minimum operating voltage even lower, saving power [52]. Furthermore, as discussed in Section 5.3, the minimum operating voltage for processing elements further in the street chain can benefit significantly from improved buffer optimization and skew management. Either channel blocks can be kept tightly coupled together, or driving buffers can be placed between long routing stretches.

**Conclusional Padding Stall Mitigation**

Integration of this hardware into a full decoding pipeline necessitates the storage of samples during the conclusional padding phase of the algorithm. The current control architecture devotes all resources to finishing the padding as fast as possible. However, this results in the first layer of SRAM to become opaque to new neural activations, when in reality this SRAM goes untouched during the majority of the conclusional padding phase. New activations writes could easily be written into SRAM as they arrive. With modification to the control sequence, a separate counter could keep track of how much of the first layer's SRAM contains valid data. Startup padding would additionally benefit from this scheme since the initially sparse computations can immediately commence rather than be stuck waiting for data to arrive.

**Control Micro-Coding**

The CNN control algorithm has several areas that could improve the efficiency of the algorithm. When the system clock is properly tuned to the data rate, MAC utilization is maximized. However, in situations that require low latency, the MACs spend a lot of time waiting for data. A configurable control would allow the system to adjust to these different scenarios and allow convolutions to commence with data already present, even before a full stride of data has arrived, improving the overall utilization of the architecture. Updates to the overall algorithmic performance like this, the stall effect mitigation discussed in the prior section, and others, all are unobtainable with the current architecture since its behavior is hard-coded into finite-state machines. While this choice was made to reduce the minimum power of low-channel count situations, improving the flexibility of the control system would not only allow for algorithmic improvements, but improve the generalization of the system to other workloads.

Replacing the generation of controls from hard-coded FSM with a small micro controller or ARM core, would allow implementation of not just FENet, but PLSR and linear decoding without major redesign of the system. Moreover, integrating this system into an SOC would almost certainly require a central processing core. Therefore, the 'additional' overhead would be minimal.

### Model Optimization and Reapplication

There are a nearly uncountable number of FENet model configurations supported by the ASIC architecture designed in this dissertation with widely varying levels of power demands. Expanding the search of the FENet model space may show improved accuracy and latency on existing hardware without sacrificing decoding accuracy. The power models presented in this dissertation offer a higher accuracy estimation of the power demands for a given model, allowing the model hyperparameters to be tuned with concrete understanding of how each parameter adjustment will affect the power requirements of the system. These estimations can be integrated into the wandb optimization framework to track the power as model parameters are explored. Adding this metric into the wandb framework would allow automated model accuracy-power optimization.

Additionally, these models were only ever trained within the context of brain-machine interfaces. Ultrasound signal processing is another 1D massively parallel signal processing workload that may find benefit in application of FENet models. Radar and sonar systems have already shown that discrete wavelet transform power for characterizations of echoes, clutter, and Doppler shift can improve performance [60]. Using the data-driven methods of FENet, may prove beneficial to systems that want to incorporate end-to-end machine-learning for a given application to maximize performance. This hardware architecture provides a basis for these decoding pipelines.

## 6.4 Lessons Learned

Throughout this journey, I was fortunate to take an algorithmic concept and build it from the ground up to ultimately integrating my architecture into a real-time neural decoding pipeline. There were many skills that I needed to develop to ensure the success of this project including a range of technical and non-technical adaptations to the needs of the situation. On the technical side, I learned not only how to take a concept from an abstract idea to an architecture, layout, and ultimately a fully integrated system, but also how to learn independently how to digest technical literature, datasheet, scour forums and documentation, and decipher the foreign languages of error logs. Learning how to learn and problem solve on my own was the most fruitful experience I could have asked for from my PhD.

**Resilience**

Multi-year projects are the staple of a dissertation. Within each technical hurdle and project timeline exists a number of opportunities to hone the skills of project management. Of these opportunities, I learned how to estimate the potential pitfalls and hurdles that a project could potentially encounter, and I learned ways to plan ahead for these pitfalls. When a pitfall was not foreseen, for instance, when I placed by IO pads far too close together for the wire-bonder to reliably bond to pads without creating shorts, I found ways to mitigate its challenges. In this case, I laser-machined chip-carriers with trace patterns to allow independent bonds to be made with one perpendicular with the pad and the trace, and another connecting the trace to the breakout pin. This experience showed me that with any road bump in a project, clever solutions can be manifest by isolating the root cause of an issue and thinking creatively.

Besides the tool usage, system design, and experience, this project also gave me the opportunity to learn resilience in the face of failure. Above all, it showed me that I am capable of doing what feels like the impossible. While at the beginning of this endeavor, I felt there was no way I would be able to learn everything I needed to know to bring the system together. I would feel a deep anxiety when I needed to start a new phase of the project that I knew nothing about. This caused me to procrastinate on the phase, hiding in the realm of the known for as long as possible. I now know how to forge into unknown waters. The state of confusion that this brings is no longer freight full, but comforting, as it is a sign of new skills to come.

**Communication**

Most importantly, I have learned how to be a better communicator of my ideas. Prior to this journey, I often felt overwhelmed with what I felt I needed to communicate. What my experience taught me was the importance of breaking down ideas into abstract components that are much easier to digest by my audience, then dissecting these ideas into their constituents. I still have more to learn in this department, but with every passing experience, I am becoming more and more able to communicate the ideas that I learn.

**Literature Review**

Literature review is one of the most important skills one develops as a PhD student, but what they don't tell you is that not everyone is the same in the way they learn and absorb information. Don't try and force yourself to learn the way you expect you should. For instance, early in my career, when I would try to read papers, I would go to the library, sit down with a paper, and try to understand every single sentence that was written. Often this would lead to countless rabbit holes chasing concepts outside my field that were entirely unimportant for me to understand the paper. Furthermore, this would take a huge amount of time and effort, by the time I would reach the conclusion section, I would be so burned out, I would barely understand the section I was reading.

What I have learned is that the best way to approach learning a new subject on my own is to start by learning the higher order concepts, hierarchically. This means starting with the most general concepts of the topic, and avoid worrying about the details until you have fully grasped the hierarchy above it. This is where review papers come in. Review papers to a remarkable job of breaking concepts into general groups, and highlighting the important information of those groups. Once you have an idea of what is generally going on, and the motivation of all the different prior art that is trying to address your problem, then, and only then is it a good idea to read the individual papers that are most relevant to your issue.

**Burnout**

You are going to get burned out, it's a fact of life in a PhD. What is important is recognizing the burnout, and know when you might be too exhausted to continue working on a particular area. When you are burned out, you will find yourself thinking only of finishing the project, and not what will help you later on. Sometimes the complexity you know you need to add to your design to make it shine might feel like it will endlessly delay your project, but it is better to push through these worries, rather than opt for simplicity just because it will finish the project.

**Criticism: It Is There to Make You Better**

It is unfortunate that graduate school occurs before you have fully formed as a person, with a complete understanding of yourself and others. Sometimes, the pressures of a PhD are what help refine your personality. The difficulties of research provide the grit and tribulations which demand that you polish out the rough edges of who you are. In my case, I took criticism far too personally. Sometimes it is hard not too, but what one needs to always remember is that criticism is not there to say you are bad, it is there to say you have an opportunity to do better. There will always be things you can do better, without criticism, you won't always know what it is you are missing.

Don't panic, don't be afraid. These reactions get you no where. Ask for help early, ask for help often. If you receive criticism that you feel is unfair, don't respond defensively, don't ignore it later. Ask for what you believe can be done to improve. It's not about you, really, it is about finding areas in your life where you can learn everywhere.

# BIBLIOGRAPHY


[1] Faraz Akram, Hee Sok Han, and Tae Seong Kim. "A P300-based brain computer interface system for words typing." In: *Computers in Biology and Medicine* 45.1 (Feb. 2014), pp. 118–125. ISSN: 00104825. DOI: 10.1016/j.compbiomed.2013.12.001.

[2] Bilal Alsallakh et al. "Mind the Pad – CNNs can Develop Blind Spots." In: (Oct. 2020). URL: http://arxiv.org/abs/2010.02178.

[3] Hyochan An et al. "A Power-Efficient Brain-Machine Interface System with a Sub-mw Feature Extraction and Decoding ASIC Demonstrated in Nonhuman Primates." In: *IEEE Transactions on Biomedical Circuits and Systems* 16.3 (June 2022), pp. 395–408. ISSN: 19409990. DOI: 10.1109/TBCAS.2022.3175926.

[4] David A. Bjånes et al. "Quantifying physical degradation alongside recording and stimulation performance of 980 intracortical microelectrodes chronically implanted in three humans for 956-2130 days." In: *Acta Biomaterialia* 198 (May 2025), pp. 188–206. ISSN: 18787568. DOI: 10.1016/j.actbio.2025.02.030.

[5] Chad E. Bouton et al. "Restoring cortical control of functional movement in a human with quadriplegia." In: *Nature* 533 (Apr. 2016), pp. 247–250. ISSN: 14764687. DOI: 10.1038/nature17435.

[6] Peter Brunner et al. "Rapid communication with a "P300" matrix speller using electrocorticographic signals (ecog)." In: *Frontiers in Neuroscience* FEB (2011). ISSN: 16624548. DOI: 10.3389/fnins.2011.00005.

[7] Alessio P. Buccino et al. "Spikeinterface, a unified framework for spike sorting." In: *eLife* 9 (Oct. 2020), pp. 1–24. ISSN: 2050084X. DOI: 10.7554/eLife.61834.

[8] Autumn J. Bullard et al. "Design and testing of a 96-channel neural interface module for the Networked Neuroprosthesis system." In: *Bioelectronic Medicine* 5.1 (Dec. 2019). DOI: 10.1186/s42234-019-0019-x.

[9] György Buzsáki. "Large-scale recording of neuronal ensembles." In: *Nature Neuroscience* 7.5 (May 2004), pp. 446–451. ISSN: 10976256. DOI: 10.1038/nn1233.

[10] Jose M. Carmena et al. "Learning to control a brain-machine interface for reaching and grasping by primates." In: *PLoS Biology* 1.2 (2003). ISSN: 15449173. DOI: 10.1371/journal.pbio.0000042.

[11] Anantha P. Chandrakasan. *Ultra Low Power Digital Signal Processing*. Tech. rep. Bangalore, Jan. 1996. DOI: 10.1109/ICVD.1996.489634.

[12] Yingping Chen et al. “An Online-Spike-Sorting IC Using Unsupervised Geometry-Aware OSort Clustering for Efficient Embedded Neural-Signal Processing.” In: *IEEE Journal of Solid-State Circuits* 58.11 (Nov. 2023), pp. 2990–3002. ISSN: 1558173X. DOI: 10.1109/JSSC.2023.3303675.

[13] Yu Hsin Chen et al. “Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices.” In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.2 (June 2019), pp. 292–308. ISSN: 21563365. DOI: 10.1109/JETCAS.2019.2910232.

[14] Yu Hsin Chen et al. “Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks.” In: *IEEE Journal of Solid-State Circuits* 52.1 (Jan. 2017), pp. 127–138. ISSN: 00189200. DOI: 10.1109/JSSC.2016.2616357.

[15] Joon Hwan Choi, Hae Kyung Jung, and Taejeong Kim. “A new action potential detector using the MTEO and its effects on spike sorting systems at low signal-to-noise ratios.” In: *IEEE Transactions on Biomedical Engineering* 53.4 (Apr. 2006), pp. 738–746. ISSN: 00189294. DOI: 10.1109/TBME.2006.870239.

[16] Alexandre Défossez et al. “Decoding speech perception from non-invasive brain recordings.” In: *Nature Machine Intelligence* 5.10 (Oct. 2023), pp. 1097–1107. ISSN: 25225839. DOI: 10.1038/s42256-023-00714-5.

[17] Zidong Du et al. “ShiDianNao: Shifting vision processing closer to the sensor.” In: *Proceedings - International Symposium on Computer Architecture*. Vol. 13-17-June-2015. Institute of Electrical and Electronics Engineers Inc., June 2015, pp. 92–104. ISBN: 9781450334020. DOI: 10.1145/2749469.2750389.

[18] Morgan Ferguson et al. “A Critical Review of Microelectrode Arrays and Strategies for Improving Neural Interfaces.” In: *Advanced Healthcare Materials* 8.19 (Oct. 2019). ISSN: 21922659. DOI: 10.1002/adhm.201900558.

[19] George W. Fraser et al. “Control of a brain-computer interface without spike sorting.” In: *Journal of Neural Engineering* 6.5 (2009). ISSN: 17412560. DOI: 10.1088/1741-2560/6/5/055004.

[20] Karunesh Ganguly and Jose M. Carmena. “Emergence of a stable cortical map for neuroprosthetic control.” In: *PLoS Biology* 7.7 (July 2009). ISSN: 15449173. DOI: 10.1371/journal.pbio.1000153.

[21] Sarah Gibson, Jack W. Judy, and Dejan Marković. “Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction.” In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18.5 (Oct. 2010), pp. 469–478. ISSN: 15344320. DOI: 10.1109/TNSRE.2010.2051683.

[22] Benyamin Haghi et al. “Enhanced control of a brain–computer interface by tetraplegic participants via neural-network-mediated feature extraction.” In: *Nature Biomedical Engineering* (2024). ISSN: 2157846X. DOI: 10.1038/s41551-024-01297-1.

[23] Leigh R. Hochberg et al. “Neuronal ensemble control of prosthetic devices by a human with tetraplegia.” In: *Nature* 442.7099 (July 2006), pp. 164–171. ISSN: 00280836. DOI: 10.1038/nature04970.

[24] Leigh R. Hochberg et al. “Reach and grasp by people with tetraplegia using a neurally controlled robotic arm.” In: *Nature* 485.7398 (May 2012), pp. 372–375. ISSN: 00280836. DOI: 10.1038/nature11076.

[25] Zichen Hu, Zhining Zhou, and Hongming Lyu. “A Microwatt/Channel Neural Signal Processor for High-Channel-Count Spike Detection and Sorting.” In: *Proceedings - IEEE International Symposium on Circuits and Systems*. Institute of Electrical and Electronics Engineers Inc., 2024. ISBN: 9798350330991. DOI: 10.1109/ISCAS58744.2024.10558215.

[26] Leonardo Iannucci et al. “Changes Over Time in the Electrode/Brain Interface Impedance: An Ex-Vivo Study.” In: *IEEE Transactions on Biomedical Circuits and Systems* 17.3 (June 2023), pp. 495–506. ISSN: 19409990. DOI: 10.1109/TBCAS.2023.3284691.

[27] Vikram Jain et al. “TinyVers: A Tiny Versatile System-on-Chip with State-Retentive eMRAM for ML Inference at the Extreme Edge.” In: *IEEE Journal of Solid-State Circuits* 58.8 (Aug. 2023), pp. 2360–2371. ISSN: 1558173X. DOI: 10.1109/JSSC.2023.3236566.

[28] Byung Hyung Kim, Minho Kim, and Sungho Jo. “Quadcopter flight control using a low-cost hybrid interface with EEG-based classification and eye tracking.” In: *Computers in Biology and Medicine* 51 (Aug. 2014), pp. 82–92. ISSN: 18790534. DOI: 10.1016/j.compbiomed.2014.04.020.

[29] Mikhail A Lebedev and Miguel A L Nicolelis. “BRAIN MACHINE INTERFACES: FROM BASIC SCIENCE TO NEUROPROSTHESES AND NEUROREHABILITATION.” In: *Physiol Rev* 97 (2017), pp. 767–837. DOI: 10.1152/physrev.00027.2016.-Brain-machine. URL: www.prv.org.

[30] Mikhail A. Lebedev and Miguel A.L. Nicolelis. *Brain machine interfaces: past, present and future*. Sept. 2006. DOI: 10.1016/j.tins.2006.07.004.

[31] Mikhail A. Lebedev et al. “Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface.” In: *Journal of Neuroscience* 25.19 (May 2005), pp. 4681–4693. ISSN: 02706474. DOI: 10.1523/JNEUROSCI.4088-04.2005.

[32] Han Sol Lee et al. “A Multi-Channel Neural Recording System With Neural Spike Scan and Adaptive Electrode Selection for High-Density Neural Interface.” In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 70.7

(July 2023), pp. 2844–2857. ISSN: 15580806. DOI: 10.1109/TCSI.2023.3268686.

[33] Yang-Guo Li et al. *Ultra Low Power High Sensitivity Spike Detectors Based on Modified Nonlinear Energy Operator*. IEEE, May 2013. ISBN: 9781467357623. DOI: 10.1109/ISCAS.2013.6571801.

[34] Johnson Loh and Tobias Gemmeke. "Dataflow Optimizations in a Sub-uW Data-Driven TCN Accelerator for Continuous ECG Monitoring." In: *2022 IEEE Nordic Circuits and Systems Conference, NORCAS 2022 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2022. ISBN: 9798350345506. DOI: 10.1109/NorCAS57515.2022.9934591.

[35] Kip A. Ludwig et al. "Using a common average reference to improve cortical neuron recordings from microelectrode arrays." In: *Journal of Neurophysiology* 101.3 (Mar. 2009), pp. 1679–1689. ISSN: 00223077. DOI: 10.1152/jn.90989.2008.

[36] Lorenzo Martini et al. "Neuronal Spike Shapes (NSS): A straightforward approach to investigate heterogeneity in neuronal excitability states." In: *Computers in Biology and Medicine* 168 (Jan. 2024). ISSN: 18790534. DOI: 10.1016/j.compbiomed.2023.107783.

[37] Nicolas Y. Masse et al. "Non-causal spike filtering improves decoding of movement intention for intracortical BCIs." In: *Journal of Neuroscience Methods* 236 (Oct. 2014), pp. 58–67. ISSN: 1872678X. DOI: 10.1016/j.jneumeth.2014.08.004.

[38] Melody M. Moore. "Real-world applications for brain-computer interface technology." In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 11.2 (June 2003), pp. 162–165. ISSN: 15344320. DOI: 10.1109/TNSRE.2003.814433.

[39] Sam Musallam et al. *Cognitive Control Signals for Neural Prosthetics*. Tech. rep. 9. 2002, p. 1. URL: www.sciencemag.org/cgi/content/full/305/5681/254/.

[40] Samuel R. Nason et al. "A low-power band of neuronal spiking activity dominated by local single units improves the performance of brain–machine interfaces." In: *Nature Biomedical Engineering* 4.10 (Oct. 2020), pp. 973–983. ISSN: 2157846X. DOI: 10.1038/s41551-020-0591-0.

[41] Mahdi Nekoui and Amir M. Sodagar. "Spike Compression through Selective Downsampling and Piecewise Curve Fitting Dedicated to Neural Recording Brain Implants." In: *BioCAS 2022 - IEEE Biomedical Circuits and Systems Conference: Intelligent Biomedical Systems for a Better Future, Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 50–54. ISBN: 9781665469173. DOI: 10.1109/BioCAS54905.2022.9948580.

[42] Angshuman Parashar et al. “SCNN: An accelerator for compressed-sparse convolutional neural networks.” In: *Proceedings - International Symposium on Computer Architecture*. Vol. Part F128643. Institute of Electrical and Electronics Engineers Inc., June 2017, pp. 27–40. ISBN: 9781450348928. DOI: 10.1145/3079856.3080254.

[43] Sankaranarayani Rajangam et al. “Wireless cortical brain-machine interface for whole-body navigation in primates.” In: *Scientific Reports* 6 (2016). ISSN: 20452322. DOI: 10.1038/srep22170.

[44] Behzad Razavi. *Design of analog CMOS integrated circuits*. McGraw Hill Education, 2017.

[45] Adam G. Rouse and Marc H. Schieber. “Spatiotemporal distribution of location and object effects in primary motor cortex neurons during reach-to-grasp.” In: *Journal of Neuroscience* 36.41 (Oct. 2016), pp. 10640–10653. ISSN: 15292401. DOI: 10.1523/JNEUROSCI.1716-16.2016.

[46] Muhammad Tariq Sadiq et al. “Exploiting pretrained CNN models for the development of an EEG-based robust BCI framework.” In: *Computers in Biology and Medicine* 143 (Apr. 2022). ISSN: 18790534. DOI: 10.1016/j.compbiomed.2022.105242.

[47] Joseph W. Salatino et al. “Glial responses to implanted electrodes in the brain.” In: *Nature Biomedical Engineering* 1.11 (Nov. 2017), pp. 862–877. ISSN: 2157846X. DOI: 10.1038/s41551-017-0154-1.

[48] Gopal Santhanam et al. “A high-performance brain-computer interface.” In: *Nature* 442.7099 (July 2006), pp. 195–198. ISSN: 14764687. DOI: 10.1038/nature04968.

[49] Hicham Semmaoui et al. “Setting adaptive spike detection threshold for smoothed TEO based on robust statistics theory.” In: *IEEE Transactions on Biomedical Engineering* 59.2 (Feb. 2012), pp. 474–482. ISSN: 00189294. DOI: 10.1109/TBME.2011.2174992.

[50] Mohammad Ali Shaeri and Amir M. Sodagar. “Data Transformation in the Processing of Neuronal Signals: A Powerful Tool to Illuminate Informative Contents.” In: *IEEE Reviews in Biomedical Engineering* 16 (2023), pp. 611–626. ISSN: 19411189. DOI: 10.1109/RBME.2022.3151340.

[51] Mohammad Ali Shaeri et al. “A 2.46-mm2 Miniaturized Brain-Machine Interface (MiBMI) Enabling 31-Class Brain-to-Text Decoding.” In: *IEEE Journal of Solid-State Circuits* 59.11 (2024), pp. 3566–3579. ISSN: 1558173X. DOI: 10.1109/JSSC.2024.3443254.

[52] Jaehyeong Sim, Somin Lee, and Lee Sup Kim. “An Energy-Efficient Deep Convolutional Neural Network Inference Processor with Enhanced Output Stationary Dataflow in 65-nm CMOS.” In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.1 (Jan. 2020), pp. 87–100. ISSN: 15579999. DOI: 10.1109/TVLSI.2019.2935251.

[53] Caleb Sponheim et al. “Longevity and reliability of chronic unit recordings using the Utah, intracortical multi-electrode arrays.” In: *Journal of Neural Engineering* 18.6 (Dec. 2021). ISSN: 17412552. DOI: 10.1088/1741-2552/ac3eaf.

[54] Eran Stark and Moshe Abeles. “Predicting movement from multiunit activity.” In: *Journal of Neuroscience* 27.31 (Aug. 2007), pp. 8387–8394. ISSN: 02706474. DOI: 10.1523/JNEUROSCI.1321-07.2007.

[55] Ian H. Stevenson and Konrad P. Kording. “How advances in neural recording affect data analysis.” In: *Nature Neuroscience*. Vol. 14. 2. Feb. 2011, pp. 139–142. DOI: 10.1038/nn.2731.

[56] Aaron Stillmaker, Zhibin Xiao, and Bevan Baas. *Toward More Accurate Scaling Estimates of CMOS Circuits from 180 nm to 22 nm*. Tech. rep. VLSI Computation Lab, ECE Department, University of California, Davis, Dec. 2011. URL: http://www.ece.ucdavis.edu/cerl/techreports/2011-4/.

[57] Daniel Valencia and Amirhossein Alimohammad. “A Real-Time Spike Sorting System Using Parallel OSort Clustering.” In: *IEEE Transactions on Biomedical Circuits and Systems* 13.6 (Dec. 2019), pp. 1700–1713. ISSN: 19409990. DOI: 10.1109/TBCAS.2019.2947618.

[58] Daniel Valencia, Patrick P. Mercier, and Amir Alimohammad. “Efficient in Vivo Neural Signal Compression Using an Autoencoder-Based Neural Network.” In: *IEEE Transactions on Biomedical Circuits and Systems* 18.3 (June 2024), pp. 691–701. ISSN: 19409990. DOI: 10.1109/TBCAS.2024.3359994.

[59] Mohan Vishwanath. *The Recursive Pyramid Algorithm for the Discrete Wavelet Transform*. Tech. rep. 3. IEEE, Mar. 1994, pp. 1603–1611. DOI: 10.1109/78.277863.

[60] Marta Walenczykowska, Adam Kawalec, and Ksawery Krenc. “An Application of Analytic Wavelet Transform and Convolutional Neural Network for Radar Intrapulse Modulation Recognition.” In: *Sensors* 23.4 (Feb. 2023). ISSN: 14248220. DOI: 10.3390/s23041986.

[61] Chengxuan Wang et al. “EWS: An Energy-Efficient CNN Accelerator with Enhanced Weight Stationary Dataflow.” In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 71.7 (2024), pp. 3478–3482. ISSN: 15583791. DOI: 10.1109/TCSII.2024.3359511.

[62] Jonathan R Wolpaw and Dennis J Mcfarland. *Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans*. Tech. rep. 2004. URL: www.pnas.orgcgidoi10.1073pnas.0403504101.

[63] Seyed Mohammad Ali Zeinolabedin et al. "A 16-Channel Fully Configurable Neural SoC With 1.52 $\mu$W/Ch Signal Acquisition, 2.79 $\mu$W/Ch Real-Time Spike Classifier, and 1.79 TOPS/W Deep Neural Network Accelerator in 22 nm FDSOI." In: *IEEE Transactions on Biomedical Circuits and Systems* 16.1 (Feb. 2022), pp. 94–107. ISSN: 19409990. DOI: 10.1109/TBCAS.2022.3142987.

[64] Carey Y. Zhang et al. "Partially Mixed Selectivity in Human Posterior Parietal Association Cortex." In: *Neuron* 95.3 (Aug. 2017), pp. 697–708. ISSN: 10974199. DOI: 10.1016/j.neuron.2017.06.040.

[65] Chen Zhang et al. "An Energy-Efficient Configurable 1-D CNN-Based Multi-Lead ECG Classification Coprocessor for Wearable Cardiac Monitoring Devices." In: *IEEE Transactions on Biomedical Circuits and Systems* 19.2 (Apr. 2025), pp. 317–331. ISSN: 19409990. DOI: 10.1109/TBCAS.2025.3530790.

[66] Tianyi Zhang et al. *QPyTorch: A Low-Precision Arithmetic Simulation Framework*. 2019. arXiv: 1910.04540 [cs.LG].

[67] Zheng Zhang and Timothy G. Constandinou. "Adaptive spike detection and hardware optimization towards autonomous, high-channel-count BMIs." In: *Journal of Neuroscience Methods* 354 (Apr. 2021). ISSN: 1872678X. DOI: 10.1016/j.jneumeth.2021.109103.

[68] Zheng Zhang et al. "Calibration-Free and Hardware-Efficient Neural Spike Detection for Brain Machine Interfaces." In: *IEEE Transactions on Biomedical Circuits and Systems* 17.4 (Aug. 2023), pp. 725–740. ISSN: 19409990. DOI: 10.1109/TBCAS.2023.3278531.

[69] Zhining Zhou, Zichen Hu, and Hongming Lyu. "A 0.53-$\mu$W/channel calibration-free spike detection IC with 98.8-%-accuracy based on stationary wavelet transforms and Teager energy operators." In: *Journal of Neural Engineering* 22.2 (Apr. 2025). ISSN: 17412552. DOI: 10.1088/1741-2552/adb5c4.