Low power smart sensor circuits for biomedical applications : applications to neural interfaces

Yao, Enyi

2016

Yao, E. (2016). Low power smart sensor circuits for biomedical applications : applications to neural interfaces. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/67221>

<https://doi.org/10.32657/10356/67221>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**LOW POWER SMART SENSOR CIRCUITS FOR
BIOMEDICAL APPLICATIONS: APPLICATIONS
TO NEURAL INTERFACES**

**YAO ENYI
SCHOOL OF ELECTRICAL AND ELECTRONIC
ENGINEERING
2016**

**LOW POWER SMART SENSOR CIRCUITS FOR
BIOMEDICAL APPLICATIONS: APPLICATIONS
TO NEURAL INTERFACES**

YAO ENYI

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2016

ACKNOWLEDGEMENTS

Finally, I got this opportunity to express my sincere thanks to the people who have helped me in the past four years of research life.

Foremost, I would like to express my gratitude to my supervisor, Assistant Professor Arindam Basu, for his continuous teaching, intelligent guidance and invaluable support during my entire period of study and research. He is one of the most smartest person I have met, with a wide spread of knowledge, great enthusiasm in the research and a sense of humour. From him, I have learned how to grow as a researcher as well as an engineer.

I would also like to thank Chen Yi, Sun Shuo, Dinup Sukumaranlab, Subhrajit Roy and Roshan Gopalakrishnan and all the other members in the BRAIN group. It is a great honour for me to work together with them for various research projects. I also want to thank all my friends in VIRTUS for their help, encouragement and support when I have any problem in my study, research and daily life.

I would also like to thank Miss Quek Gan Siew Kim, Miss Lim Tan Gek Eng, Mr David Robert Neubronner, Mr Sia Liang Poo and all the other technical and administration staff in VIRTUS, Centre of Excellence in IC Design, for their help in the software and testing equipments.

I would also want to thank for the scholarship provided by the EDB, Singapore, the UMC 65-nm IC Shuttle Project for the chip fabrication facility provided by MediaTek, chip soldering and testing facilities and equipments provided by VIRTUS, Centre of Excellence in IC Design and the travelling grant to attend international conferences overseas provided by School of Electrical and Electronic Engineering.

Last but not the least, I would like to express my thanks to my parents for giving a life to

me, and my sister for the spiritual support throughout my life.

ABSTRACT

Health problem is one of the most important issues for our human race. In the past a few decades, smart sensors have been widely used in the healthcare area for real-time monitoring, diseases diagnosis and prevention, and a lot of research activities have been carried on in the design of low-cost, innovative biomedical smart sensors. Benefiting from the advances in the submicron complementary metal oxide semiconductor (CMOS) technologies, highly integrated smart sensor system have been developed for different applications. A smart sensor is an intelligent system which can be used to detect or predict problems. Typically it is composed of some basic blocks with function of signal acquisition, feature extraction and classification. This thesis focuses on the design of some basic circuit blocks in the smart sensor system for biomedical applications. We take the neural signal recording as an example, present a novel circuit design on the spike signal detection, feature extraction and classification.

In this thesis, we proposed a novel low power, compact, current-mode spike detector (SPD) with feature extractor for real-time neural recording systems where neural spikes or action potential (AP)s are of interest. Such a circuit can enable massive compression of data facilitating wireless transmission. This design can generate a high signal-to-noise ratio (SNR) output by approximating the popularly used nonlinear energy operator (NEO) through standard analog blocks. We show that a low pass filter after the NEO can be used for two functions - (i) estimate and cancel low frequency interference and (ii) estimate threshold for spike detection. The circuit is implemented in a 65 nm CMOS process and occupies $200\text{ }\mu\text{m} \times 150\text{ }\mu\text{m}$ of chip area. Operating from a 0.7 V power supply, it consumes about 30 nW of static power and 7 nW of dynamic power for 100 Hz input spike rate making it the lowest power consuming spike detector reported so far.

After the detection, feature extraction is needed to get the neural spike shape information,

which can be utilized for spike sorting. In this thesis, a spike based feature extraction method was proposed based on the structure of current controlled oscillator (CCO). This feature extractor can generate a number of digital spike pulses according to the input neural spike signal. These generated spike pulses will be counted by an asynchronous counter every $300\ \mu\text{s}$. These counting values form the spike features which contain enough information to discriminate between different shapes of recorded spike waveforms. The classification engine we are using for spike sorting in this thesis is K-means clustering algorithm. The advantage of the spike based feature extraction is that the feature spikes can be directly utilized for neural signal decoding. In typical implementations, the machine learner for spike sorting is followed by a separate machine learner acting on spike trains at the output of spike sorter. This second machine learner decodes the intention of the subject. Hence if we can use a spike based feature extractor, it may be possible to use a single machine learner to directly decode intentions without going through the separate step of spike sorting. This is the motivation behind our choice of spike based feature extraction. In this work, we do not show the intention decoding but just show that there is sufficient information available in the spike based features to do spike sorting. Hence, only one machine learner based classifier is enough for neural recording.

This thesis also proposed a hardware implementation of extreme learning machine (ELM) which can perform the local processing in smart sensors while dissipating very low power. This ELM can be applied in the smart sensor for different applications. The proposed circuit utilizes device mismatch prevalent in today's VLSI process to perform a significant part of the computation while a digital back end enables precision in the final output. The particular machine learning algorithm we use is ELM. Mismatch in silicon spiking neurons and synapses are used to perform the vector-matrix multiplication that forms the first stage of this classifier and is the most computationally intensive. System simulations and measurement have been conducted to evaluate the dependence of performance (in a classification and a regression task) on analog and digital parameters like weight resolution, maximum spike frequency etc. It is shown that the proposed implementation is more energy efficient as opposed to custom digital implementations for a classification task. In order to verify its function, some real-world bench

mark binary classification datasets have been employed in the measurement showing that the performance of our design is comparable with recent publications and software simulations of other machine learning system. This system was implemented in a $0.35\ \mu\text{m}$ CMOS process which can operate from 0.8 V to 3.3 V power supply with a lowest classification energy 0.47 nJ/op and maximum classification speed 518 MMAC/s.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
ABSTRACT	iii
LIST OF FIGURES	x
LIST OF TABLES	xvii
LIST OF ABBREVIATIONS	xviii
CHAPTER	
1. Introduction and Motivation	1
1.1 Background	1
1.2 Neural Recording System	2
1.3 Motivation	3
1.4 Contributions	5
1.5 Organization	6
2. Background and Literature Review	8
2.1 Microelectrode Array	8
2.2 Neural Amplifier	9
2.3 Spike Detection	10
2.3.1 Overview	10
2.3.2 Adaptive Detection	11
2.3.3 Nonlinear Energy Operator	13
2.4 Feature Extraction and Spike Sorting	15
2.5 Hardware Implemented Machine Learner	22
2.6 Summary	25
3. A 0.7 V, 40 nW Compact, Current-Mode Neural Spike Detector in 65 nm CMOS	27
3.1 Background	27

3.2	System Architecture and Modes of Operation	30
3.2.1	System Architecture	30
3.2.2	Operation Mode I: Rejection of Low Frequency Interference and Noise (LFIN)	37
3.2.3	Operation Mode II: Spike rate independent Automatic Threshold setting	40
3.3	Power Dissipation Model	44
3.4	Measurement Results	45
3.5	Automatic Threshold Setting	52
3.6	Detection Probability Estimator	53
3.7	Summary	56
4.	Spike based Feature Extraction for Combined Spike Sorting and Intention Decoding	61
4.1	Background	61
4.2	Objectives	62
4.3	System Architecture	63
4.3.1	Differentiator	64
4.3.2	Current Rectifier	65
4.3.3	Current-Controlled Oscillator (CCO)	66
4.4	Algorithm: pFSDE Features	67
4.5	Results	70
4.5.1	Simulation	70
4.5.2	Measurement	72
4.6	Summary	74
5.	VLSI Extreme Learning Machine: A Design Space Exploration	75
5.1	Background	75
5.2	Objectives	76
5.3	ELM Theory	78
5.4	System Architecture	80
5.4.1	Input Generation Circuit (IGC)	81
5.4.2	Neuron	83
5.4.3	Current Mirror Array	87
5.4.4	Parameter Choice	89
5.5	Noise, Speed and Energy dissipation	92
5.5.1	Noise	92
5.5.2	Speed	93
5.5.3	Energy	97
5.6	Input Dimension and Hidden Layer Extension Technique	99
5.7	Measurement Results	103
5.7.1	Characterization	103
5.7.2	Speed and Power	106
5.7.3	Regression and Classification	108

5.7.4	Dimension Increase With Weight Reuse Technique	109
5.7.5	Comparison	110
5.7.6	Robustness	110
5.8	Summary	114
6.	Conclusion and Future Work	116
6.1	Conclusion	116
6.2	Future Work	118
APPENDICES		121
A.	Author's Publication	122
A.1	Journal Papers	122
A.2	Conference Papers	122
BIBLIOGRAPHY		123

LIST OF FIGURES

Figure

1.1	Architecture of the conventional neural recording system.	2
1.2	Architecture of the proposed neural recording system.	3
2.1	(a) Michigan Probe [1,2]; (b) Utah microelectrode array [3].	9
2.2	Schematic of the capacitive feedback neural amplifier [4].	10
2.3	Log-log plot of gain with increased frequency for the amplifier [5].	10
2.4	Overview of spike detection and sorting procedure.	11
2.5	Block diagram of the adaptive spike detection algorithm [5].	12
2.6	If we feed the neural signal to a comparator and set the reference the same value of the standard deviation σ , the comparator will have an output waveform with a duty cycle of 0.159 [5].	13
2.7	Pseudo-resister based adaptive threshold circuit [6].	13
2.8	Block diagram of the NEO based spike detector [6].	14
2.9	Schematic of the peak detector [6].	16
2.10	System diagram for the feature extractor for informative samples algorithm [7].	17
2.11	Overview of the procedure for automatic spike sorting. (A) Neural spikes are detected adaptively. (B) A number of wavelet coefficients is selected for relevant features of the spikes. (C) Using the superparamagnetic clustering algorithm to classify these spikes [8].	18
2.12	System architecture of the spike sorting DSP [9].	19

2.13	Illustration of NEO threshold crossing and maximum-derivative alignment on a sample spike waveform [9].	20
2.14	Support vector machine acceleratory (SVMA) configurable via memory-mapped registers to enable various classifier structures and kernels. The FSM controls the configurable data path unit (DPU) to realize the computations. [10]. . . .	21
2.15	Parallel architecture for the multi-class SVM classifier [11].	23
2.16	Multiply accumulate cell (MAC) and reference cell implemented in the MVM [11].	24
2.17	Analog deep learning engine architecture and possible application scenarios [12].	25
2.18	(a) The architecture. The execution procedure for the algorithm is presented in (b) to (e): (b) classification, (c) training load, (d) memory write and (e) belief construction [12].	26
3.1	Neural recording system with real-time spike detection circuit. Only the spike detection circuit with the architecture as shown in the dashed box is implemented in this work.	28
3.2	Neural spike activity and background noise with a threshold to be set between the noise and signal.	29
3.3	Block diagram of the spike detector architecture; transistor level circuit of each sub-block are shown later. In the fabricated chip, threshold is set offline. The circuit in dashed lines are for mode II which are not included in the chip but are simulated to show performance achievable for automatic setting of threshold.	30
3.4	The principle for a simpler estimation of NEO by considering the effect of NEO on sine wave inputs.	31
3.5	Simulated signal path: (a) Input spike signal; (b) Output current from the differentiator I_{in} ; (c) Input current I_1 and (d) output current I_2 of the current squaring block; (e) Output current I_{LPF1} from LPF1 and (f) Final detection output.	32
3.6	Proposed NEO approximation compared to the typical NEO when 3 different spike signals are applied as the inputs.	34
3.7	Circuit diagram of the current rectifier that uses a feedback amplifier (FA) to pin the input node.	35

3.8	Detailed circuit of the feedback amplifier (FA).	35
3.9	Structure of the (a) current squaring circuit and (b) the low pass filter, both employing translinear loop principle in their design.	36
3.10	Choice of f_1 depends on average spike width. (a) Sample spike signal and (b) corresponding waveform of $I_{LPF1} - I_{LPF2}$ for different values of f_1 in spice simulation.	37
3.11	In mode I, adding an extra pseudoresistor based filter to create the traditional approximation to threshold can easily be done.	38
3.12	(a) Input spike waveform. (b) If f_2 is small, input spike of smaller amplitude can go undetected. (c) Increasing f_2 allows for detection of small amplitude spike since I_{LPF2} reduces to baseline quickly.	39
3.13	Variation of background estimate with input spike firing rate for noise standard deviations of (a) 0.05. (b) 0.15 and (c) 0.35 respectively normalized to input spike amplitude. Clearly, the conventional estimate based on low value of f_2 increases a lot for high firing rates. The effect is more prominent for cases with low noise.	40
3.14	Flowchart of algorithm for automatically setting threshold based on I_{LPF2} . . .	43
3.15	An example of operation of the above algorithm describing the timing of events.	43
3.16	Layout and die photo of the prototype fabricated in 65 nm CMOS.	45
3.17	Experimental setup for the measurement of prososed SPD.	46
3.18	(a) Segment of the input simulated signal. (b) Measured output pulses of the chip showing time of detection.	47
3.19	(a) Original spike signal that is superimposed on low frequency interference. The SPD can correctly detect spikes even when the amplitude of low frequency interference is (b) same as or (c) 8X larger than spike amplitude. . . .	48
3.20	Measured performance of the spike detection in mode I compared with other reported techniques. (a) Probability of missing detection and (b) Probability of false alarm with different noise level.	49
3.21	Simulated performance of the spike detection in mode II compared with mode I and conventional automatic threshold setting. (a) Probability of missing detection and (b) Probability of false alarm with different noise level.	50

3.22	Automatic threshold detection for (a) high SNR input signal and (b) low SNR input signal with conventional setting of $f_2 = 2$ Hz can miss spikes with small amplitude if coming after a big amplitude spike while setting $f_2 = 64$ Hz results in successful detection. (c) FNR for the previous case where the amplitude of the smaller spike is varied and SNR is high.	51
3.23	Possible circuit implementation of the algorithm for automatic threshold setting described in Fig. 3.14.	52
3.24	Schematic of the oscillator based DPE circuit.	54
3.25	Input neural signal, detection and oscillation outputs.	55
3.26	The DPE output feature distributions for low noise level.	56
3.27	The FNR and FPR plots for low noise level with DPE.	57
3.28	The DPE output feature distributions for high noise level.	58
3.29	The FNR and FPR plots for high noise level with DPE.	59
4.1	(a) The flowchart for the conventional neural recording system (b) The flowchart for the proposed neural recording system.	62
4.2	Block diagram of the feature extractor. The differentiator and the current rectifier can be shared with the spike detector block described in Chapter 3.	63
4.3	Circuit diagram of the current rectifier that uses a feedback amplifier (FA) to pin the input node and (b) detailed circuit of the feedback amplifier (FA).	64
4.4	Detailed circuit of the feedback amplifier (FA).	64
4.5	The schematic of the CCO.	65
4.6	The oscillation waveform of the CCO.	66
4.7	The three template spike shapes with a noise level of 0.05 and the corresponding output waveforms of the two CCOs.	67
4.8	Input neural signal and the outputs coming from CCO1 and CCO2.	69
4.9	Comparison of PCA (for 3 and 10 PC's) with pFSDE performance in spike clustering. First we can see that FSDE works better for white noise than the noise added in the dataset comprising similar spike shapes. Second, simulations and chip measurements show FSDE and pFSDE works better than PCA for high noise conditions.	70

4.10	Variation of classification error for pFSDE with pulse-count per spike for varying background noise levels. Note optimal pulse-count (corresponding to a specific threshold) for each curve in the plot.	71
4.11	Microphotograph of the feature-extractor IC.	72
4.12	Input neural signal and the measured outputs coming from CCO1 and CCO2.	73
5.1	The architecture of ELM algorithm. d is the dimension of the input data and L denotes the number of hidden layer neurons.	78
5.2	System architecture of the mixed signal integrated circuit that implements the first stage of ELM; the second stage is implemented in digital domain. The digital input data is converted to current by the IGC and then multiplied by random weights w_{ij} in the current mirror array. The current is converted to digital domain by the combination of a spiking neuron and a counter.	80
5.3	Timing diagram of the ELM system where RN_in is a global reset, $Data_in$, A and CLK_in are SPI control signals to transfer input data to the IC. NEU_EN enables the neuron to produce spikes while CLK_cnt is used to read out the counter values C one by one.	81
5.4	Schematic of input generation circuit (IGC) for one channel. A reference current is split according to the 10 bits of input data to create I_{DAC} . The capacitor C ensures sufficient SNR when the current is mirrored to the L columns. An active current mirror is enabled to allow fast settling when I_{DAC} is small.	82
5.5	Schematic of the neuronal oscillator circuit followed by an asynchronous counter. The neuron is enabled when control signal NEU_EN is high. The capacitors can be digitally reconfigured and have the following values: $C_{a1} = 100\text{fF}$, $C_{a2} = 200\text{fF}$, $C_{b1} = 50\text{fF}$, $C_{b2} = 100\text{fF}$	83
5.6	Oscillation waveforms at different nodes of the neuron circuit.	84
5.7	(a) Neuron spiking frequency initially increases with the increase of the input current I^z till $I^z = I_{flx}$. It then reduces and becomes zero finally when $I^z = I_{rst}$. (b) The transfer function (solid line) of the neuron with input I^z and output H can be saturated at a pre-defined value of 2^b by stopping the counter.	85
5.8	Comparison of neuron spiking frequency between theory and simulation in SPICE show close match.	86
5.9	Simulated neuron spiking frequency with increasing input current for 3 different VDD. The curves saturate at higher maximum frequencies for higher VDD. Note the logarithmic scales for both plots.	87

5.10	Simplified circuit diagram of one current mirror for noise analysis.	88
5.11	Variations of L_{min} with I_{sat}^z/I_{max}^z show that the optimal value of this ratio is ≈ 0.75	90
5.12	Variations of classification accuracy with the resolution of output weight β showing 10 bits is sufficient for accurate classification.	91
5.13	Variations of classification accuracy with the number of bits of counter output H demonstrating that $b \approx 6$ is enough for optimal performance. Each of the curves are averaged over 50 trials.	92
5.14	Using active current mirror for small input currents can boost the bandwidth by $5.84X$	94
5.15	Variation of neuron counting time (T_{neu}) and current mirror settling time (T_{cm}) reduce as maximum input current per dimension (I_{max}) is increased. Further, T_{neu} increases exponentially with increase in b	95
5.16	Contours where T_{cm} is equal to T_{neu} in the space of counter dynamic range 2^b and input dimension d . For increasing d , the total current input to a neuron I^z keeps on increasing thus increasing oscillation frequency. Hence, it can support higher dynamic range 2^b in the same time T_{neu}	96
5.17	Variation of energy per classification operation (E_c) with varying maximum value of input current I_{max}^z for three different settings of VDD.	99
5.18	The same plot as in (a) but replacing I_{max}^z with its corresponding T_{neu} from equation (5.19).	100
5.19	The extension from a 2×3 random projection matrix to 6×6 by weight reuse technique	101
5.20	(a) Schematic of peripheral circuit for hidden layer extension by shifting the input data stored in the registers and (b) its timing diagram.	102
5.21	(a) Schematic of circuit for input dimension extension by shifting and summing the output counter values and (b) its timing diagram.	103
5.22	Die photo of the prototype chip fabricated in $0.35\mu\text{m}$ CMOS.	104
5.23	Measured transfer function of hidden layer neurons when the digital input varying from 0 to 1023 with $d = 1$ and $T_{neu} = 10\text{ms}$	105

5.24	A surface plot showing the mismatch in weights of the 128×128 current mirror synapses. The output counter values for different neurons are plotted for $T_{neu} = 10\text{ms}$ when $Data_in = 100$ is set on each input channel one by one.	106
5.25	Histogram showing the log-normal distribution of the input weights obtained from 5.24 for the 128×128 current mirror array.	107
5.26	Regression of underlying sinc function (in blue) based on a set of noisy samples (in green).	108
5.27	Comparison of hidden layer outputs for three different values of VDD in (a) the conventional case and (b) normalized case. The normalization results in less variation of output due to change in VDD.	112
5.28	Comparison of performance when normalized and non-normalized hidden layer outputs are used for classification of (a) Australian credit and (b) Bright-data sets from the UCI repository.	113

LIST OF TABLES

Table

2.1	Comparison of Different Methods for Spike Detection.	14
2.2	Comparison of Different Algorithms for Hardware Machine Learner	24
3.1	Transistor Sizes.	33
3.2	Power dissipation in biasing block.	45
3.3	Table comparing this design with the state-of-the-art.	53
3.4	Current Status for Each Block	58
4.1	Comparison Table	73
5.1	Comparison of Different Methods for Hardware Machine Learner.	76
5.2	Chip Summary	104
5.3	Mean and standard deviation of ΔV_{Tn}	105
5.4	Measured performance on Binary Classification Datasets from UCI repository	109
5.5	Comparison Table	111
5.6	<i>Sinc</i> function regression using normalized h_j	114

LIST OF ABBREVIATIONS

AP	action potential
NEO	nonlinear energy operator
ED	energy detection
SNR	signal-to-noise ratio
CMOS	complementary metal oxide semiconductor
ELM	extreme learning machine
BMI	brain-machine interface
MEA	micro-electrodes array
ADC	analog-to-digital converter
DAC	digital-to-analog converter
LPF	low pass filter
VGA	variable gain amplifier
CCO	current controlled oscillator
MLCP	machine learning co-processor
MEMS	microelectromechanical systems
SPD	spike detector
DPE	detection probability estimator
SVM	support vector machine
MVM	matrix vector multiplier
MAC	multiply accumulate cell
DML	deep machine learning
FGM	floating gate transistors

Chapter 1

Introduction and Motivation

1.1 Background

Over the past a few decades, smart sensors for biomedical application have become increasingly popular and dramatic progress has been observed in this field. These sensors can be utilized in the diagnosis of the diseases, or even can be used to help people to monitor their health conditions in their daily lives. For example, the epilepsy is a kind of neural disorder with recurrent evoking of seizures, which is a very common disease and significantly affects the daily lives of the patients. Almost 0.7% of the population suffer from this kind of disease [13]. It cost over 15 billion in Europe in 2004 [14]. Typically, the monitoring of seizures always happens in the hospital. People have to take time off work and stay in the hospital with heavy equipments connected to their heads, which is quite inconvenient. It is also inaccurate as the monitoring time is generally not long enough and the people under monitoring are kept away from their daily lives where these neural conditions usually happen. However if these seizures appear when the patients are not in the hospital, it would be very dangerous for them. According to [15], we know that generally an electrical onset of seizure occurs around 7.5 second before clinical onset. If this electrical onset signal can be observed, certain medical treatment can be applied to prevent or suppress the coming clinical seizure onset. Hence, smart sensors which can be applied to monitor and predict the seizures for the people with epilepsy are quite important.

Figure 1.1: Architecture of the conventional neural recording system.

The rapid development in the microelectronics industry and the availability of advanced CMOS fabrication technology down to the micro- and nanoscale have made the design of a variety of novel biomedical smart sensor systems possible. By integration of local signal processing unit and data communication block, these biomedical sensors can be utilized for real-time monitoring, diagnosis and even prevention of many diseases.

1.2 Neural Recording System

Among all the parts of our human body, the brain is the least understood and the disease related to the nervous system is always the most difficult to treat. Such big challenges in this area attract scientists and engineers to focus their research to develop novel devices to understand and unlock the secrets of neural coding. One particular example is the neural recording system. The neural signal recording is quite significant for both research and diagnosis purposes. Direct recordings from the neurons have enabled the doctors and researchers to know what is happening in the brain. Our brain is quite a complicated system with over 100 billion neurons connected with each other by the synapses. By using the high-density micro-electrode arrays, the activities of a large number of neurons can be captured simultaneously. These information can be used for the diagnosis or prostheses. For example the brain-machine interface (BMI) is a direct interface between the brain and the electronic and mechanical devices [16–18]. Through a BMI system, neural signal can be utilized to help the patients to restore sensory or motor function. For example, the information derived from the motor cortex can be used to control a prosthetic artificial arm or generate stimulation pulses to the brain to alleviate pain or reduce

Figure 1.2: Architecture of the proposed neural recording system.

the epileptic seizures.

Figure 1.1 shows an example of conventional neural recording system, which comprises an analog front-end to sense the neural signal in the brain, an analog-to-digital converter (ADC) to convert the analog signal into digital, spike detection and two machine learner based digital processor: one is for spike sorting, the other one is used for decoding. The aim of spike detection is to locate the neural spike signal among the background noise. Then spike sorting process is to cluster all these spike detected into different groups. Each group belongs to one particular neuron. After that, intention decoding will extract any useful information behind these spikes to help us understand what our brain is thinking. Recently, fully implanted neural recording has gained more and more popularity in neural science research due to its high accuracy. With the rapid development of high-density micro-electrodes array (MEA), over several hundreds of recording channels can be integrated in a single chip. A wireless data transmitter is also needed for communication between the sensor and a computer. Most of the designs usually transmit the digitalized data outside the human body to a computer or a central station for processing. After processing, the corresponding control signal will be generated and sent back to the stimulator or some other prosthetic device for disease treatment or body function repairment.

1.3 Motivation

As described in the background, smart sensors can be utilized for biomedical applications to improve the health condition of patients. The development in today's CMOS technology provides advantages for us to design the next generation smart sensor systems. These sensors

are portable, or even implantable, capable for long term monitoring and early detection of abnormalities. Hence, the health condition of the patients can be observed even at home. They do not need to stay in the hospital with bulky and complicated instruments connected for a long time. What's more, the signal processing unit also has a possibility to predict any abnormalities so that a further treatment can be conducted. For example, implantable or wearable neural recordings can be used to help to monitor the brain activity for the patients with epilepsy. Once a seizure is detected, medicine or simulations can be given to release the seizures.

Usually, the sensory part should be implantable under the skull, record the neural signal, and transmit the neural raw data outside wirelessly to avoid tissue infections. Then a signal processing unit outside our body will derive useful information from these spike signals and generate the corresponding control parameters for some robotic prosthetic device or a stimulator in the brain, forming a close-loop feedback. Since the battery can not be replaced very frequently, the energy for these circuits is better to be derived from energy harvesting or wireless power transmission. Power dissipation budgets are very small for each circuit. Moreover, energy efficiency is required to avoid the damage to the brain tissue caused by the heat [19]. Consider the case where we want to simultaneously transmit 100 channels of data sampled at 10 bit resolution and sampling rate of 20 kHz; the resulting data rate of 20 Mbps is beyond the capability of current low power transmitters. So given the limited power budget available to these sensors, the need for wireless data transfer will finally restrict the number of simultaneously recording channels as well as the precision of detection. One way to solve this problem is to use smart sensor with local processing which will lower the transmission data rate resulting in lower power consumption if the local circuits dissipate minimal power. For neural recording, since spikes are sparsely distributed, possible approach to compression for applications requiring only spike timing information is to just transmit a '1' when there is a spike and '0' for other times. The long stream of zeros can be effectively compressed by schemes like run-length encoding resulting in huge compression ratios of 50 – 5000 depending on spike frequency at input. Fig. 1.2 shows a proposed architecture of neural recording system with spike detection, spike sorting and decoding integrated together.

Motivated from the above analysis, in this thesis, a low power, low voltage, analog SPD based on an approximation to NEO is proposed for real time neural recording. Such a circuit can enable massive compression of data facilitating wireless transmission. A spike based feature extractor is also proposed for spike sorting. The advantage of this spike based feature extraction is that the feature spikes can be directly utilized for neural signal decoding. We also proposed a novel design of a feature extractor which could be able to get enough information for both spike sorting and decoding. Additionally, a low power neuromorphic machine learner that can perform the local processing in smart sensors while dissipating very low power is also investigated in this thesis.

1.4 Contributions

This thesis focuses on the circuit design for biomedical smart sensors. The ELM can be employed in the smart sensors as a co-processor which has been demonstrated to be more compact and have lower power consumption. For a specific application, the proposed implementation of ELM has been applied in the neural recording system for action potential detection and sorting. By exploring novel designs for a smart sensor with high performance and minimized power dissipation, a number of contributions has been achieved:

- A novel low power, compact, current-mode spike detector circuit for real-time neural recording systems is designed in a 65 nm process. Such a circuit can enable massive compression of data facilitating wireless transmission. This design can generate a high signal-to-noise ratio (SNR) output by approximating the popularly used nonlinear energy operator (NEO) through standard analog blocks.
- To reduce sensitivity of threshold setting for the spike detector, a current-mode oscillator based detection probability estimator (DPE) is proposed to reject false positives caused by the background noise.
- A pulse based feature extraction is proposed to get the neural spike shape information for

spike sorting. The extracted feature spikes can also be directly utilized for neural signal decoding.

- A hardware implementation of ELM that can perform the local processing in smart sensors while dissipating very low power was proposed. The proposed circuit utilizes device mismatch prevalent in today's VLSI process to perform a significant part of the computation while a digital back end enables precision in the final output. Mismatch in silicon spiking neurons and synapses are used to perform the vector-matrix multiplication that forms the first stage of this classifier and is the most computationally intensive.
- A prototype for the ELM classifier is fabricated in a $0.35\ \mu\text{m}$ process. Different datasets are used for the evaluation of the performance. A design space trade-off between speed and power is explored and it is shown that an optimal operation frequency exists for a given value of VDD.
- Weight rotation technique is proposed to extend the physical limitation of the hardware neural network. By rotation of the input weight, we can extend the input dimension and hidden layer to a wider range.

1.5 Organization

This thesis is organized as follows. Chapter 1 introduces the background knowledge for biomedical smart sensors and an overview of this thesis including motivations for doing research in this area, as well as the major contributions of this thesis.

Chapter 2 reviews recently published smart sensor circuits for biomedical applications. In particular, we take the neural recording as an example. Recent publications on the design of analog front-end, neural action potential detection and feature extraction are presented. Publications on machine learner circuits are also reviewed. For ELM, basic explanation of the algorithm is also provided.

Chapter 3 describe a novel low power, compact, current-mode spike detector circuit for real-time neural recording systems where neural spikes or action potentials (AP) are of interest.

Such a circuit can enable massive compression of data facilitating wireless transmission. This design can generate a high signal-to-noise ratio (SNR) output by approximating the popularly used nonlinear energy operator (NEO) through standard analog blocks. We show that a low pass filter after the NEO can be used for two functions-(i) estimate and cancel low frequency interference and (ii) estimate threshold for spike detection. To reduce sensitivity of threshold setting, a current-mode oscillator based detection probability estimator (DPE) is proposed to reject false positives caused by the background noise.

Chapter 4 proposed a novel method to extract pulse based features of the spike signal. These features contain enough information to do spike sorting as well as spike decoding. In this way, we can combine the 2 different machine learner to one for both spike sorting and intention decoding.

Chapter 5 presents a compact low-power, high performance hardware implementation of the extreme learning machine (ELM) which can be utilized as a co-processor in the biomedical smart sensors. Mismatch in silicon spiking neurons and synapses are used to perform the vector-matrix multiplication that forms the first stage of this classifier and is the most computationally intensive. A mixed-signal machine learning IC is implemented which can be configured for a wide range of applications. Various types of datasets are applied to the system for evaluation of the function. A design space trade-off between speed, power and accuracy is explored. Some techniques to improve performance in VLSI random projection networks is also proposed in this chapter. A technique to overcome the hardware limitation for input dimension and hidden layer is also proposed.

Finally, the summary of my research completed during this 4 years of PhD program is presented in Chapter 6. Future works for the design of low power smart sensor circuits for biomedical applications are also discussed and illustrated in this chapter.

Chapter 2

Background and Literature Review

In this chapter, a literature review is provided to introduce various aspects of smart sensors, especially for neural recording. Typically, a neural recording system is consisting of an MEA and neural signal conditioning circuits which is implanted under the skull. During neural recording, the weak neural signal in the brain can be acquired, amplified and transmitted wirelessly in the digital format. Useful information can be decoded from the neural signal for scientific or clinical applications. To improve the performance of smart sensor system, literature on the design of hardware implementation of machine learner are also reviewed. These machine learners can be utilized for neural signal processing as well.

2.1 Microelectrode Array

MEA is the very foundation for neural recording. During the past few decades, with the advances in the microelectromechanical systems (MEMS) technology, tens to hundreds of electrodes can be positioned in array with very small dimension for chronic intracortical recording and stimulation. Fig. 2.1 shows two types of MEAs which are very popular used in the neural recording. Fig. 2.1(a) illustrates a 256-site microelectrode array with 4 probes 3-D microassembled together [1]. The original design of this microelectrode is 2-D with 64-site developed by the University of Michigan. Lithography technology is utilized to define the probes in the array. Typical dimension for the probe is 3 mm in length, 50 μm in width and 15 μm in thickness for the silicon shank. Fig. 2.1(b) shows a silicon based three-dimensional electrode array which is

Figure 2.1: (a) Michigan Probe [1,2]; (b) Utah microelectrode array [3].

developed by the University of Utah [3]. The size for monocrystalline substrate is $4.2 \times 4.2 \times 0.12 \text{ mm}^3$ with 100 conductive needles on it for neural signal acquisition. Each of the needle is 0.09 mm thick, 1.5 mm long and covered with platinum for better electrical conductivity.

2.2 Neural Amplifier

For modern neural recording, the MEA is usually utilized to record the signal from a number of neurons in the brain. In order to get enough information for clinical prosthetic or control, signal from multiple neurons is required. Typically, the MEA is placed extracellularly with a few tens of microns away to detect a signal with 50-500 μV amplitude. Hence, an amplifier is usually required to amplify the signal, reject the dc offset and eliminate the noise.

[4] proposes a low-power and low-noise amplifier design for neural recording, shown in Fig. 2.2. The capacitors C_1 and C_2 decide the midband gain of the amplifier A_m , which is equal to C_1/C_2 . MOS-bipolar pseudo-resistor elements $M_a - M_d$ are utilized in the feedback for their very large impedance R_2 , which will create a very low high-pass corner to eliminate the DC offset and the low frequency noise. Frequency response of this amplifier is shown in Fig. 2.3, where the lower and upper cut-off frequencies are illustrated. The lower cut-off frequency f_L is set by the value of input capacitor C_1 and R_2 , while the upper cut-off frequency f_H is determined by the transconductance of the OTA G_m , load capacitance C_L and the midband gain A_m .

Figure 2.2: Schematic of the capacitive feedback neural amplifier [4] .

Figure 2.3: Log-log plot of gain with increased frequency for the amplifier [5] .

2.3 Spike Detection

2.3.1 Overview

As mentioned in previous sections, action potentials contain the information encoded in the brain. For most scientific and prosthetic applications, only relevant spike timing information for every neuron is needed for signal decoding. Therefore, spike detection is required for each neural recording sensors. It is one of the most basic procedures in the neural recordings. It lays the foundation for spike sorting and decoding. On the other hand, the data rate need to be compressed for wireless transmission. In this way, we do not need to transmit the detailed

Figure 2.4: Overview of spike detection and sorting procedure.

10 to 12 bits digitalized neural signal all the time. Since spikes are sparsely distributed in the neural signal, one possible approach to compression the data is to just transmit a ‘1’ when there is a spike and ‘0’ for other times. The long stream of zeros can be effectively compressed by schemes like run-length encoding resulting in huge compression ratios of 50 – 5000 depending on spike frequency at input.

Fig. 2.4 shows an overview procedure of spike detection and sorting. Spike detection is to identify spikes from the very noisy background with the chip. Spike amplitudes vary widely between different electrodes. Besides, background noise generated by distant neurons and electronic noise may also cause a lot of false detections. These factors cause difficulties and challenges for the detection. The original and simplest method for spike detection is amplitude thresholding. The threshold can be manually generated by a digital-to-analog converter (DAC). An ADC is utilized for the user to observe the neural signal and update the threshold value [20] [5].

2.3.2 Adaptive Detection

Instead of manually setting the threshold, we hope the threshold level could be adaptively set with best performance. In general, the threshold can be set to be several times of the background noise level. Since the background noise from actual neural recordings has a Gaussian distribution and if we can measure its standard deviation σ , the threshold could be set to several times of σ [5]. Fig. 2.5 shows an implementation to measure the standard deviation σ of background noise and adaptively detect the spike [21]. As shown in Fig. 2.6, if the threshold value

Figure 2.5: Block diagram of the adaptive spike detection algorithm [5].

is set to be just the same value of the standard deviation σ , the probability for the neural signal to exceed the threshold is 0.159. Hence, if we use the standard deviation σ as the reference of a comparator, it will generate an output waveform with a duty cycle of 0.159. According to this observation, a feedback loop can be utilized to adjust the reference voltage of the Comparator A to be just the same value of σ . Multiplied this voltage by a factor of N , we can get the final threshold for the detection.

The standard deviation can also be estimated based on the median, which leads to a threshold to be calculated as

$$\begin{cases} Thr = 4\sigma_n; \\ \sigma_n = median\left\{\frac{|x|}{0.6745}\right\}, \end{cases} \quad (2.1)$$

where x is the bandpass-filtered neural signal and σ_n is the estimation of standard deviation of the background noise [22]. Some other designs utilize a pseudo-resistor based low-pass-filter for the estimation of the background noise, as is shown in Fig. 2.7. The pseudo-resistor can be realized by the anti-parallel diodes, which could provide a very large impedance [4]. The cut-off frequency is set to be around 1-2 Hz. The input signal will be low-pass filtered and doubled to the output through a current mirror formed by MN3 and MN4. Hence, only when the input

Figure 2.6: If we feed the neural signal to a comparator and set the reference the same value of the standard deviation σ , the comparator will have an output waveform with a duty cycle of 0.159 [5].

signal is larger than twice of the average background noise, a detection can be generated.

Figure 2.7: Pseudo-resistor based adaptive threshold circuit [6].

2.3.3 Nonlinear Energy Operator

Among various existing action potential detection algorithm, the nonlinear energy operator (NEO) gains more and more popularity in the publications of recent years due to its relatively

Table 2.1: Comparison of Different Methods for Spike Detection.

Method	Advantages	Disadvantages	Reference
Threshold	Simple implementation	Bad performance at low SNR	[5, 23]
NEO	Good performance at low SNR	Difficulty for the implementation of high order derivative; Spike rate dependent noise estimate	[6, 24, 25]
EC-PC	Good performance; Spike rate dependent noise estimate	Large power and area	[26]

good performance at low SNR. NEO is defined as:

$$NEO(x) = \left(\frac{dx}{dt}\right)^2 - \frac{d^2x}{dt^2} \cdot x. \quad (2.2)$$

Many new hardware designs based on NEO have been proposed in recent publications. The block diagram for one of them is represented in the Fig. 2.8 [6]. The differentiation is generated by a gm-C differentiator, while the multiplication is performed by Gilbert cell. The final subtraction is generated by the differential current outputs.

We compare different algorithms for hardware implementation of neural spike detection, which is shown in Table 2.1.

Figure 2.8: Block diagram of the NEO based spike detector [6].

2.4 Feature Extraction and Spike Sorting

Feature extraction is an important step in the process of neural recording. The function of feature extraction is to get rid of the similar information among different classes of spikes and get their unique characteristics. Feature extraction is often conducted after spike detection. Typically, the extracted features will be utilized to assign each of the detected spike to the individual neuron. We call this process ‘spike sorting’. The basic procedure for spike sorting typically include: (1) spike detection, (2) extraction of distinctive features from different spike shapes, and (3) classification of the spikes according to these features. Spike sorting methods are typically based on clustering predefined spike shape features such as peak-to-peak amplitude, width, or principle components [27] [28].

A variety of algorithms or circuits have been developed to extract the features of neural spikes. [6] presents a circuit design to capture the extreme positive and negative values of neural spike. The schematic of the positive peak detector is shown in Fig. 2.9, and the negative detector has a same schematic with opposite polarity. From this figure, we can understand that the differential pair is used to charge capacitor C_1 . When the voltage of $V_{SG,MP1}$ is less than 0, the subthreshold current in $MP1$ would be minimized and the output of the peak detector will drift to VDD during periods with little activity. When both of the positive and negative peak values are detected, an ADC followed by will turn on to digitize the peak values of the spike. At the same time, *Hold* signal is pull up to high to force the input to 0, preventing the output from changing during the analog-to-digital conversion. After the conversion is completed, *Hold* signal goes to low and reset the output to 0.

[7] describes a feature extractor implementation utilizing informative samples algorithm. The corresponding system architecture is shown in Fig. 2.10. To quantify the information contained in each spike samples, the probability density for $s - th$ sample point of N spikes can be estimated by a kernel density [29]. Circuit in Fig. 2.10 is just utilized to calculate the kernel density. If $x_1, x_2, \dots, x_M$ are independent and identically distributed of a random variable, to

Figure 2.9: Schematic of the peak detector [6].

approximate the probability density function, the equation to calculate the kernel density is

$$f(x) = \frac{1}{h^d} \sum G\left(\frac{x-x_j}{h}\right), \quad (2.3)$$

where h is called bandwidth which is a smoothing parameter and d is the data space dimension, $G()$ is an arbitrary isotropic kernel function, a square function is utilized in the hardware implementation. One of the solution is to estimate local bandwidth with a pilot kernel density estimation, which is

$$f_0(x) = \frac{1}{h_0^d} \sum_{j \neq i} G\left(\frac{x_i - x_j}{h_0}\right), \quad (2.4)$$

where h_0 is the initial value for the global bandwidth. According to the pilot density function, we can get the local bandwidths

$$h_{x_i} = h_0 \left[\frac{\lambda}{f_0(x_i)} \right]^{0.5}, \quad (2.5)$$

where λ is a constant number, which can be obtained by taking the geometric mean of $f(x_i)|_{i=1, \dots, N}$.

Replace bandwidths with equation 2.5, the final density estimation is

$$f(x) = \frac{1}{h_{x_j}^d} \sum G\left(\frac{x-x_j}{h_{x_j}}\right), \quad (2.6)$$

The final results are positive and negative peaks which can be utilized to classify neural spikes.

As shown in the figure, the multiplexer on the top is used to choose between the initial h_0 and

the estimated h_j . The calculation of $G()$ and $f(x)$ are sharing one single divider through another multiplexer. The SRAM is employed for the storage of the calculation results for the probability density function.

Figure 2.10: System diagram for the feature extractor for informative samples algorithm [7].

[30] introduced a spike sorting algorithm using wavelets and superparamagnetic clustering. The procedure for spike sorting is illustrated in Fig. 2.11. The wavelet transform is a time-frequency decomposition process of the signal with optimal resolution for both the time and frequency domains. Additionally, it releases the requirement of signal stationarity. It is the convolution between the input signal $x(t)$ and the wavelet function $\psi_{a,b}(t)$,

$$W_{\psi}X(a,b) = \langle x(t) | \psi_{a,b}(t) \rangle, \quad (2.7)$$

where $\psi_{a,b}(t)$ can be obtained by shifting the unique wavelet function $\psi(t)$,

$$\psi_{a,b}(t) = |a|^{-\frac{1}{2}} \psi\left(\frac{t-b}{a}\right), \quad (2.8)$$

where a and b are the scale and translation parameters. The wavelet transform maps the signal

with only one independent variable t to a function with two independent variables a and b . By correlating the input signal with wavelet functions of different sizes, several scales of signal in detail can be obtained.

Figure 2.11: Overview of the procedure for automatic spike sorting. (A) Neural spikes are detected adaptively. (B) A number of wavelet coefficients is selected for relevant features of the spikes. (C) Using the superparamagnetic clustering algorithm to classify these spikes [8].

The theory of superparamagnetic clustering is described in [8]. The first step of this algorithm is to illustrate the m selected features for each spike i by a point x_i in an m -dimensional

phase space. The interaction between points x_i can be represented as

$$J_{ij} = \begin{cases} \frac{1}{K} \exp(-\frac{\|x_i - x_j\|^2}{2a^2}) & \text{if } x_i \text{ is a nearest neighbor of } x_j \\ 0 & \text{else} \end{cases} \quad (2.9)$$

where K is the number of nearest neighbors and a is the average nearest-neighbors distance. In the next step, they assign an initial random state s from 1 to q to each point x_i . After that, N Monte Carlo iterations will be run for various temperatures T utilizing the Wolf algorithm [31] or Swendsen-Wang algorithm [32]. The probability for the nearest neighbors of x_i to change their states to s_{new} can be calculated by

$$p_{ij} = 1 - \exp(-\frac{J_{ij}}{T} \delta_{s_i, s_j}), \quad (2.10)$$

where T is the temperature. Neighbors that modify their values create a frontier and cannot modify their values at first can do so if it can be visited again during the iteration. The frontier will be updated until it does not change any more.

Figure 2.12: System architecture of the spike sorting DSP [9].

[9] presents a novel method for hardware implementation of energy-efficient spike sorting circuit. Spike detection is performed by NEO and the maximum derivative is used for spike

Figure 2.13: Illustration of NEO threshold crossing and maximum-derivative alignment on a sample spike waveform [9].

alignment [33]. System architecture of the proposed DSP core to implement the spike sorting algorithm is shown in Fig. 2.12 .

During the training process, the threshold is calculated by the threshold-calculation module, which is

$$Thr = C \frac{1}{N} \sum_{n=1}^N \Psi(n), \quad (2.11)$$

where $\Psi(n)$ is the NEO output. Reason for the number of training samples N was chosen to be a power of 2 is because the division by N can be performed just by shifting. C is a constant. In hardware implementation, an empirical value for C is 8 [34]. When $\Psi(n)$ goes over this calculated threshold, a spike event is detected. Once a spike detection event is triggered, 72 samples of data, which is 3 ms, will be saved in the on-chip memory for further calculation. One of the recorded spike waveforms is illustrated in Fig. 2.13. The point of maximum derivate (k) is calculated by the Maximum Derivative block in Fig. 2.12, which is

$$k = \arg \max_n (s(n) - s(n-1)). \quad (2.12)$$

This point also functions as the address offset to access the memory block. The aligned spike waveform is read out so that k corresponds to the 11th sample point of the aligned spike wave-

form with 48 samples. Fig. 2.13 shows the 48 samples of the aligned spike and the point with maximum derivative value. Discrete-derivative features of the aligned spike can be computed by

$$dd_{\delta}(n) = s(n) - s(n - \delta), \quad (2.13)$$

where $s(n)$ is the n -th sample point of the spike signal and the value for δ is 1, 3 and 7. The computation of discrete derivatives occupies $3 \times$ of the storage for the representation of a spike signal. However, only a small number of the features is sufficient for clustering. For each level of the extracted derivatives, seven coefficients would be selected by uniform sampling of the coefficients. Hence there would be 21 coefficients for one spike signal.

Figure 2.14: Support vector machine acceleratory (SVMA) configurable via memory-mapped registers to enable various classifier structures and kernels. The FSM controls the configurable data path unit (DPU) to realize the computations. [10].

2.5 Hardware Implemented Machine Learner

Inspired by the biological neural networks, neural network has been developed for the applications like pattern recognition, categorization, control and robotics for many years. It has a variety of advantages over conventional von Neumann computers, including massive parallelism, distributed computation, learning ability, false tolerance, etc.. With the advances in the IC design and fabrication technology, hardware implemented neural network has gained more and more attention. Hardware neural network has a faster speed over software neural network and could be integrated together with the smart sensor system as a co-processor. Among various neural networks, support vector machine (SVM) is the most popular algorithm to be implemented in the hardware [10, 11, 35–39].

In [10], a support-vector-machine accelerator realizing various classification algorithms as well as various kernel functions and kernel formulations is proposed. However, the utilization of digital memory consumes significant area and power. To overcome these problems in digital domain, an analog SVM based pattern classifier was proposed for biometric signature verification in [11]. The parallel architecture of the implemented SVM with two matrix vector multiplier (MVM)s is shown in Fig. 2.15. The input signal $x[n]$ will be fed column-parallel to MVM1. Inner-products between the input signal $x[n]$ and the template vectors x_s is computed by MVM1. In this design, a quadratic function $K(x_s, x[n]) = (x_s \cdot x)^2$ is implemented as the kernel. The inner-products generated by MVM1 will be presented to the kernel to produce the squared kernel values. The kernel values finally will row-parallel go to MVM2 for computation to get the SVM category outputs. Floating-gate transistors are utilized in the most computationally intensive part of the SVM for the nonvolatile data storage and computation. One of these multiply accumulate cell (MAC) using floating-gate transistor is shown in Fig. 2.16. Since the transistors are biased in the sub-threshold region, there would be an exponential relationship between the drain current I_{ds} and either the gate voltage V_g or the source voltage V_s [40, 41],

which can be illustrated in the following equation:

$$I_{ds} = I_o \frac{W}{L} e^{-\kappa V_g / U_T} e^{V_s / U_T}, \quad (2.14)$$

where U_T is the thermal voltage which is around 26 mV, κ is the gate coupling factor [42–44].

From Fig. 2.16 we can calculate the output current

$$I_{out} = I_{in} e^{-\kappa(V_g - V_{gref}) / U_T} = I_{in} \beta \quad (2.15)$$

Figure 2.15: Parallel architecture for the multi-class SVM classifier [11].

Another machine learner which gains more and more popularity recently is deep machine learning (DML), which is inspired by the hierarchical presentation of information in the brain to achieve robust automated feature extraction [45]. Typically, people are trying to use GPU-based platforms to implement the deep machine learning [46]. However it is power hungry and difficult to be integrated together with a biomedical smart sensor. [12] proposed a analog deep machine-learning engine, shown in Fig. 2.17. A 4-2-1 hierarchy is formed by seven identical cortical nodes which could obtain the regularities in its input signals through an unsupervised learning process. The raw data like an image goes to the first layer, which is the lowest and generate an output *belief*. The outputs of lower layers will be fed to their respective higher layers as the inputs. *Beliefs* generated from the lower layers represent local features and *beliefs* from higher layers represent global features. Dimensionality will keep decreasing all the way

Figure 2.16: Multiply accumulate cell (MAC) and reference cell implemented in the MVM [11].

along the signal path from bottom to the top. The final output of the node from the top layer will be employed for post processing due to its rich features contained.

An online k-means classification algorithm is implemented by the node, shown in Fig. 2.18. This algorithm will be able to extract the features from the input signal by recognizing spatial density patterns in the input space. The cluster which is recognized can be characterized by the centroid mean $\hat{\mu}$ and variance $\hat{\sigma}^2$, which are also called center of mass and spread. Each node is composed of an 8×4 array of reconfigurable analog computation cells (RAC), which is divided by 4 8-dimensional input centroids. floating gate transistors (FGM) are also used for the storage of the centroids' parameters $\hat{\mu}$ and $\hat{\sigma}^2$. For the input, an 8 dimensional observation vector sequence $o[n]$ goes to the RAC array in row-parallel.

Table 2.2: Comparison of Different Algorithms for Hardware Machine Learner

Algorithm	Advantages	Disadvantages	Reference
SVM	Simple structure	Moderate accuracy; iterative tuning	[10]
Deep Learning	Excellent accuracy	Computation intensive: large area and power	[47]
ELM	Fast learning; No iterative tuning	Moderate accuracy	[48]

To conclude this literature review and facilitate comparison, a table comparing different

Figure 2.17: Analog deep learning engine architecture and possible application scenarios [12].

algorithms concerning hardware implementation of machine learner is presented in Table 2.2.

2.6 Summary

This chapter has discussed the fundamental information for spike detection and sorting. Spike sorting is a significant procedure for the neural signal decoding. Before spike sorting, we have to detect the spike signal. For implantable neural recordings, spike detection could also be utilized to compress the data which is to be wirelessly transmitted outside. Recently published algorithms and circuits has been presented in this chapter. Additionally, hardware implemented machine learner circuit can be applied in the smart sensors as a co-processor, which can be used to help process the sensed signal locally. Literatures on the machine learner based sensors have been reviewed in this chapter.

Figure 2.18: (a) The architecture. The execution procedure for the algorithm is presented in (b) to (e): (b) classification, (c) training load, (d) memory write and (e) belief construction [12].

Chapter 3

A 0.7 V, 40 nW Compact, Current-Mode Neural Spike Detector in 65 nm CMOS

As mentioned in the introduction, smart sensors are very important for the development of new therapies and detection methods for life threatening diseases. To design such a smart sensor for biomedical application, we have to minimize its power dissipation for a longer battery life. To reduce the power consumption of the system, data reduction is an effective method. In this chapter, we focus on a circuit to reduce the data rate for an example case of neural recording—a neural spike detector.

3.1 Background

With the advance of today's semiconductor technology, up to several hundred channels have been integrated in the single-chip neural recording system to monitor multiple neurons' activities in the brain simultaneously. One emerging application is brain machine interfaces (BMIs) which is a direct real-time interface between brain and computers or other devices. It could be utilized to treat some conditions like Parkinson's disease, restore motor impairment or even for neuroscience research. Usually, the sensory part should be implantable under the skull, record the neurons' action potentials, also called spike signals, and transmit the neural raw data outside wirelessly to avoid tissue infections. Then a signal processing unit outside our body will derive the information from these spike signals and generate the corresponding control parameters for

Figure 3.1: Neural recording system with real-time spike detection circuit. Only the spike detection circuit with the architecture as shown in the dashed box is implemented in this work.

some robotic prosthetic device or a stimulator in the brain, forming a close-loop feedback.

Fig. 3.1 illustrates the general architecture of the neural recording system. Each channel is composed of a pre-amplifier (A), a variable gain amplifier (VGA) and an ADC while the wireless transmitter is shared by all the channels. Consider the case where we want to simultaneously transmit 100 channels of data sampled at 10 bit resolution and sampling rate of 20 kHz; the resulting data rate of 20 Mbps is beyond the capability of current low power transmitters [49]. Since spikes are sparsely distributed in the neural signal, one possible approach to compression for applications requiring only spike timing information is to just transmit a ‘1’ when there is a spike and ‘0’ for other times. The long stream of zeros can be effectively compressed by schemes like run-length encoding resulting in huge compression ratios of 50 – 5000 depending on spike frequency at input. Neural spike detector is typically integrated in every channel of the neural recording system with various requirements summarized below:

- The spike detector should be able to reject the background noise especially when the SNR is low.
- The estimation of background noise level should not be dependent on the the firing rate of the input spike.
- Automatic threshold setting is needed for a fully implanted device.

Figure 3.2: Neural spike activity and background noise with a threshold to be set between the noise and signal.

- The spike detector should have low power consumption.

The original and simplest method for spike detection is amplitude thresholding where the threshold is set to be several times of the background noise level [5], as is shown in Fig. 3.2. However, the performance of this method is known to be very sensitive to setting of threshold. As an alternative, a nonlinear energy operator (NEO) has been proposed for improved performance of spike detection. NEO is defined as:

$$NEO(x) = \left(\frac{dx}{dt} \right)^2 - \frac{d^2x}{dt^2} \cdot x. \quad (3.1)$$

Many new hardware designs based on NEO have been proposed in recent publications [6] [24] [25]. However, most of their work utilize pseudo-resistor based low pass filter (LPF) with cut-off frequency $\approx 1 - 2$ Hz to estimate the background noise level. This makes the background estimate and consequently NEO threshold sensitive to input spike firing rate. Also, they depend on accurate setting of threshold for good performance.

In this paper, a low power analog spike detector (SPD) based on an approximation to NEO is presented. Due to the use of current mode design, power supply voltage could be reduced to 0.7V resulting in lowest reported power for NEO spike detectors. Further, we show that in

Figure 3.3: Block diagram of the spike detector architecture; transistor level circuit of each sub-block are shown later. In the fabricated chip, threshold is set offline. The circuit in dashed lines are for mode II which are not included in the chip but are simulated to show performance achievable for automatic setting of threshold.

our topology, a LPF with high cut-off frequency ($\approx 60 - 100$ Hz) acting on the NEO output can be used in two ways: (i) it can be used to reject low frequency interference and power line noise—in that case threshold for NEO is determined externally and (ii) if there is a low frequency rejection filter before this SPD block, the output of this LPF can be used to create a threshold for NEO that is largely independent of input spike rate. To reduce the dependence of performance on threshold, an estimate of detection probability is also generated which is used to reduce false positives thus increasing the robustness of spike detection.

3.2 System Architecture and Modes of Operation

3.2.1 System Architecture

The schematic of the proposed spike detector is shown in Fig. 3.1 with a detailed block diagram in Fig. 3.3. It consists of a differentiator, a current rectifier, a squaring circuit, two LPFs with different cut-off frequencies and a comparator. The principle of spike detection, similar to [25], is based on an approximation to the NEO output for a sine wave input. For an input sine wave $A \sin(\omega t)$, the output of the NEO is a constant with magnitude equal to $A^2 \omega^2$. The response to a sine wave is important since a single cycle of a sine, with alternating positive and negative half cycle, resembles a spike signal coarsely. As shown in Fig. 3.4, this output can be obtained by low-pass filtering the square of the derivative of input as well. The

Figure 3.4: The principle for a simpler estimation of NEO by considering the effect of NEO on sine wave inputs.

advantage of this alternative is that it requires only a single differentiator, squarer and low-pass filter. This principle is similar to the energy detection (ED) method in [25]; however, unlike [25] who did not use a LPF after the squarer, we found it to be necessary to reject high frequency noise and also to prevent double detections for the same spike. Do note that even though Fig. 3.4 shows the response of the proposed method for a sinusoidal input, the approximation holds for other signals like neural spikes—the only difference in that case is the larger temporal spread ($\approx 1\text{ms}$) of the detection output of the proposed method compared to traditional NEO ($\approx 0.5\text{ms}$). However, this is not an issue since the typical duration of a neural spike is $\approx 2\text{ms}$ which is larger than this temporal spread. Fig. 3.5 shows the signal waveforms observed from different points along the signal path in simulation. Fig. 3.6 shows our NEO approximation compared to the typical NEO when 3 different spike signals are applied as the inputs. From these figures, we can see this NEO approximation method works well for the neural spike signals and is similar to the typical NEO—both methods selectively amplify the high frequency component of the signal more than low frequency parts.

As shown in Fig. 3.3, a capacitor C_{in} ($= 4\text{ pF}$) is used as a differentiator where the output is a current proportional to (dV_{in}/dt) . Larger values of C_{in} will produce more input current at the expense of chip area. The value of 4pF is chosen to ensure at least $\approx 1\text{nA}$ of current is created for a 50mV sinusoid with duration larger than 1ms that mimics a spike. Since the

Figure 3.5: Simulated signal path: (a) Input spike signal; (b) Output current from the differentiator I_{in} ; (c) Input current I_1 and (d) output current I_2 of the current squaring block; (e) Output current I_{LPF1} from LPF1 and (f) Final detection output.

Table 3.1: Transistor Sizes.

Devices	Size (μm)
M_p, M_n	0.08/0.06
M_{a1}, M_{a2}	10/0.5
$M_{a3}, M_{a4}, M_{a5}, M_{a6}, M_{a7}, M_{a8}$	10/3
M_1, M_3, M_4, M_5	4/2
$M_2, M_6, M_7, M_8, M_9, M_{10}, M_{11}$	8/2
$M_{12}, M_{13}, M_{14}, M_{15}, M_{16}, M_{17}$	16/2

current squaring circuit can only process uni-directional current, a class-B current mirror similar to [50] and shown in Fig. 3.7 is adopted as a rectifier before it. A negative feedback is added in the class-B current mirror through an OTA (with a bias current $I_a = 20\text{ nA}$) which reduced the input impedance by a factor equal to its open-loop gain. It also ensures that one terminal of the capacitor stays at a fixed voltage enabling it to act as a differentiator. This feedback amplifier (FA) employs long channel length transistors instead of cascodes to get high gain at low voltages. The bias current is chosen to have a bandwidth of $\approx 20\text{kHz}$ that is several times larger than the $\approx 5\text{kHz}$ bandwidth of a typical spike. A class-B current mirror has a deadzone that limits the minimum detectable input current [50]. This deadzone is directly proportional to parasitic capacitance C_{gs} of transistors M_n and M_p . To reduce this, both of these transistors are kept as minimum sized core transistors. Further, these transistors have low V_T to enable reduction of power supply. The simulated value of the minimum detectable signal at V_{in} due to this deadzone is approximately $100\mu\text{V}$. Do note that this value is at the input of the rectifier; typically there are amplifier stages preceding the SPD with gains $\approx 500 - 1000$ implying that the deadzone induced minimum detectable signal limit at the probe is around $0.1 - 0.2\mu\text{V}$.

Figure 3.9(a) details the circuit of the current squaring block. It is a log domain translinear loop circuit where all the transistors are operated in the sub-threshold region. Then the input output relation can be easily obtained as:

$$I_{out} = \frac{I_{in}^2}{I_{ref}}, \quad (3.2)$$

where I_{ref} is tunable with 8 digital control bits (using a current splitter topology as in [51])

Figure 3.6: Proposed NEO approximation compared to the typical NEO when 3 different spike signals are applied as the inputs.

with a maximum value of 0.5 nA. I_{ref} can be used as a second variable gain control or as an alternative to tune the detection quality when threshold is fixed. The output of the squarer is then low pass filtered to obtain the NEO estimate following the architecture in Fig. 3.1. Figure 3.9(b) shows the detailed circuit of a current-mode LPF which is applied in the design of the spike detector. The transfer function of the LPF is:

$$\frac{I_{out}(s)}{I_{in}(s)} = \frac{I_{b1}/I_{b2}}{1 + \tau s}, \quad (3.3)$$

where $\tau = \frac{CU_t}{\kappa I_{b2}}$, U_t denotes thermal voltage and κ is inverse of sub-threshold slope of a MOS [42]. Thick oxide transistors are employed in this design so that the cut-off frequency is not affected by gate leakage currents present in thin oxide transistors. The cut-off frequency of this

Figure 3.7: Circuit diagram of the current rectifier that uses a feedback amplifier (FA) to pin the input node.

Figure 3.8: Detailed circuit of the feedback amplifier (FA).

low pass filter can be adjusted by tuning either the bias current or the capacitance C . As can be seen from Fig. 3.3, there are two LPFs with cut-off frequencies $f_1 = 1/\tau_1$ and $f_2 = 1/\tau_2$ in the architecture. The corresponding capacitors are set as $C_1 = 2.1$ pF while C_2 can vary in the range of 3.6 – 14.3 pF. The bias currents I_{b1} and I_{b2} are equal and can be tuned with a maximum value of 0.25 nA. The range of values of C_1 and C_2 are chosen to set $f_1 \approx 400$ Hz and $f_2 \approx 60$ Hz while not lowering I_b below 200pA for enhanced reliability. LPF1 is the one mentioned in Fig. 3.4

Figure 3.9: Structure of the (a) current squaring circuit and (b) the low pass filter, both employing translinear loop principle in their design.

needed to produce an estimate of NEO whereas LPF2 might have two roles as will be discussed next. As can be seen from Fig. 3.3, the difference of the output currents of the two low pass filters provides the final output current. This is like a band-pass filter transfer function given by:

$$\begin{aligned}
 & I_{LPF1}(s) - I_{LPF2}(s) \\
 &= \frac{1}{I_{ref}} (sC_{in}V_{in}(s) * sC_{in}V_{in}(s)) \left(\frac{I_{b1}/I_{b2}}{1 + s\tau_1} - \frac{I_{b1}/I_{b2}}{1 + s\tau_2} \right)
 \end{aligned} \tag{3.4}$$

where “*” denotes convolution. The final detection output voltage of the SPD can be approximated as:

$$out = u(I_{LPF1} - I_{LPF2} - I_{thr}) \tag{3.5}$$

where $u()$ denotes the heaviside function implemented at the drain of transistor M_{17} due to the high impedance of the node. This is buffered by the inverters to produce the final output. I_{thr} is a threshold (with a maximum value of 0.25 nA) whose value is decided externally for our IC. A method for setting it automatically will be discussed later and simulation results for that case will be shown. Transistors sizes for all the circuits are shown in Table 3.1.

The choice of f_1 depends on the width of a spike signal. As shown in Fig. 3.10, if f_1 is too high ($\approx > 1kHz$), the output of LPF1 has sharp changes that can produce multiple detection pulses for the same input spike. Also, it introduces more noise from the derivative in the detection output. As a thumb rule, we use inverse of the spike width to set f_1 . Here, since spike

(a)

(b)

Figure 3.10: Choice of f_1 depends on average spike width. (a) Sample spike signal and (b) corresponding waveform of $I_{LPF1} - I_{LPF2}$ for different values of f_1 in spice simulation.

width is approximately 2.5 ms, $f_1 = 400$ Hz. The requirements for LPF2 are based on different modes of operation and is discussed next.

3.2.2 Operation Mode I: Rejection of Low Frequency Interference and Noise (LFIN)

Neural recording signals have high power in low frequency also termed as local field potentials or LFP. The amplitude of such signals are often $\approx 10X$ larger than spike signals and can hence cause problems in spike detection if not filtered. Also, there are often power line interference mixed in the neural signal which can cause false detections as well. Typically, these

Figure 3.11: In mode I, adding an extra pseudoresistor based filter to create the traditional approximation to threshold can easily be done.

signals that cause large fluctuations of the baseline need to be filtered by a band-pass filter. Here, we propose to create this band-pass operation by taking the difference of the output of the two filters where f_2 is chosen to be less than f_1 and defines the lower edge of pass band. A first-order attenuation of low frequency signals is also provided by the input capacitor C_{in} . Assuming spike signals have energy at ≈ 500 Hz and LFIN signal is at ≈ 50 Hz, we get 20 dB of de-emphasis due to C_{in} only. Thus, the proposed architecture integrates low frequency rejection operation with spike detection and removes the burden of band-pass filtering from preceding stages. Hence, we choose $f_2 \approx 60 - 100$ Hz in this case to ensure removal of low frequency components. In section 5.7, we shall present measurement results to demonstrate the ability of this circuit to reject large, low frequency disturbances.

Note that the use of LPF2 seems very similar to estimating background noise level for automatic threshold setting for spike detection [5] [6]. However, those methods use LPF with very low cut-off frequency to estimate average background activity in spike frequency band while in our case f_2 is higher to reject out of band LFIN. It can still be used for automatic threshold determination as will be shown in the next subsection; however, for this mode of operation we expect to determine threshold externally and communicate this to the implanted unit. This can be done during initial calibration phase when $S1 = S2 = 1$ in Fig. 3.1 thus sending both raw data and detection output. This incurs a maximum bandwidth overhead of 10% during calibration assuming 10-bit raw neural data samples. After calculating threshold

Figure 3.12: (a) Input spike waveform. (b) If f_2 is small, input spike of smaller amplitude can go undetected. (c) Increasing f_2 allows for detection of small amplitude spike since I_{LPF2} reduces to baseline quickly.

and other parameters (e.g. f_1, f_2), we set $S1 = 0, S2 = 1$ to go into data compression mode where only detection output is sent or raw data is sent only when a spike is detected. Moreover, if needed this threshold can easily be approximated in the traditional way as shown in Fig. 3.11. Similar to [6], we can use a pseudoresistor and an extra transistor M18 to create the traditional approximation to the threshold. The pseudoresistor forms a low pass filter having a low cut-off frequency with the gate capacitance of M18. Finally, the current through M18 can act as an estimate of threshold.

Figure 3.13: Variation of background estimate with input spike firing rate for noise standard deviations of (a) 0.05. (b) 0.15 and (c) 0.35 respectively normalized to input spike amplitude. Clearly, the conventional estimate based on low value of f_2 increases a lot for high firing rates. The effect is more prominent for cases with low noise.

3.2.3 Operation Mode II: Spike rate independent Automatic Threshold setting

As mentioned earlier, generally a pseudo-resistor based low pass filter with a very low cut-off frequency (about 0.1 – 2 Hz) is used to estimate the background noise occurring in spike frequency band due to distant neural firing activity. However, this mean value based threshold estimation suffers from a severe dependence on input spike firing rate as shown in [30]. A further problem not explored by [30] is caused when there are multiple spikes on the same electrode with different amplitudes occurring within a short time span. For example, Fig. 3.12(a) shows such a waveform with a 10X smaller spike occurring 5ms after a bigger one. Figure 3.12(b) shows spice simulation of our proposed detector circuit where $f_2 = 2$ Hz and I_{LPF2} is

treated as the background estimate. It can be seen that due to the long time constant, I_{LPF2} takes very long to reach back to the true estimate of background and remains higher than I_{LPF1} thus missing the small spike. In actual recording scenario, the situation is worse since the threshold is set to 3 – 5 times of this estimate. The solution proposed in [30] is to use median filtering instead of mean; but hardware implementations of median filtering take a lot of area due to memory resources and do not scale to multiple channels [52]. Here, we propose an alternate solution to threshold estimation by using a filter with a relatively higher cut-off frequency f_2 . The advantage can be understood intuitively from Fig. 3.12(c) where increasing f_2 to 64 Hz allows I_{LPF2} to decay back to the true background quickly and allows detection of the small spike.

To further illustrate the point, we show MATLAB simulations of how the estimation of background level increases with the spike firing rate similar to [30]. The spike signals are chosen from the database developed in [30] since it is a commonly used database in many recent publications on spike detection circuits [24] [53] [54] and provides a baseline for comparison. The database comprises 594 different average spike shapes compiled from recordings in the neo-cortex and basal ganglia. For generating background noise, spikes randomly selected from the database were superimposed at random times and amplitudes aiming to mimic the background noise of actual recordings that is generated by the activity of distant neurons. A spike train at a desired frequency (with a spike shape chosen from the same database) is then superimposed on the background; the amplitude of the spike is normalized to 1 while the background noise level is determined by its standard deviation with respect to spike amplitude. These are fed into a MATLAB model of our architecture to find average value of I_{LPF2} , the background estimate, for different values of f_2 . We also calculate the background estimate for a conventional NEO.

The simulation results of background estimate are shown in Fig. 3.13 for three different standard deviations of background noise that correspond to SNR values of 15.5 dB, 5.9 dB and -1.4 dB respectively. Each plot is normalized to the true background estimate obtained when firing rate is zero. In other words, the true background estimate is normalized to 1 in this case similar to [30]. It can be seen that for the conventionally chosen small values of

f_2 , the background estimate increases appreciably ($\approx 20X$) with input firing rate in low noise condition. On the other hand, for higher values of $f_2 \approx 64\text{Hz}$, this estimate stays almost constant till commonly observed physiological firing rates of 50 Hz [55]. Also, it can be noted from Fig. 3.13(c) that the effect of increased firing rate on conventional background estimate is not so severe in high noise cases indicating our proposed method is especially useful in low noise recording scenario. The above scheme can be incorporated as an automatic threshold estimation block in SPD units as shown in Fig. 3.1 and can eliminate the need for calibration present in the earlier mode of operation. The automatic threshold estimation block is not implemented in the chip. However, simulations have been done for the evaluation of the function. Before presenting a possible circuit implementation of this scheme, it is useful to note that this method of estimating threshold based on I_{LPF2} assumes that the large baseline variation due to LFIN has been already removed by a band pass filter before the SPD block. In other words, LPF2 cannot perform the function of rejecting low frequency interference in mode 2. Also, note that f_2 cannot be set too high—this makes the background estimate noisy. To get a better estimate, I_{LPF2} can be averaged over some time T_{avg} to reduce fluctuations. The threshold can then be set to a value k times larger (typically $k \approx 2 - 5$) to reduce the probability of false detections. In this work, we have optimized the value of k over the dataset. First, an optimal value of threshold that minimizes error is found by sweeping the threshold value. Then a first order fit is performed with to find a linear relation between the background estimate and the optimal threshold. The slope of this fit is used as k . Note this value is constant for all noise levels. In practice, this value may be found based on some initial recording data and can be thought of as an optional calibration step for enhanced performance.

A detailed flow chart describing the algorithm to set I_{thr} based on I_{LPF2} is shown in Fig. 3.14 and a possible circuit implementation is described in the appendix. There are two parallel threads—the one on the left *creates* $\overline{I_{LPF2}}$ by averaging I_{LPF2} every T_{avg} seconds while the one on the right decides *when* to update I_{thr} with I_{avg} , a variable that keeps track of the last non-corrupt value of $\overline{I_{LPF2}}$. This is necessary since the value of $\overline{I_{LPF2}}$ in one period is corrupted if there was a spike detected in that averaging period. Moreover, since we know that I_{avg} will be higher than

Figure 3.14: Flowchart of algorithm for automatically setting threshold based on I_{LPF2} .

Figure 3.15: An example of operation of the above algorithm describing the timing of events.

true background for sometime after the occurrence of a spike, we do not update I_{thr} with the value of I_{avg} all the time. Instead, we update I_{thr} only when a spike is detected—at that time, the value stored in the variable I_{avg} is the one computed in the time step before the current spike detection has happened and is hence not corrupted.

Figure 3.15 depicts the operation of the above algorithm in time domain. It can be seen that the value of $\overline{I_{LPF2}}$ always rises to a high value in the clock period following the occurrence of a spike in the input. However, I_{avg} does not change to this corrupted average value—it is updated to the value of $\overline{I_{LPF2}}$ in earlier clock period as shown by the arrows.

3.3 Power Dissipation Model

The power dissipation for the proposed SPD is composed of two parts: static and dynamic. The static power is mostly consumed by the biasing blocks. The breakdown of the power dissipated in each block from spice simulations is illustrated in 3.2. Among these elements, there is about 322 nW power consumption for the main biasing block which can be amortized across all the channels on the chip. It will split the large current generated from a bandgap or bootstrap circuit into sub-micro level current to be utilized by the each block. For each channel, there are also some local current splitters and the total power dissipation for that is 29.4 nW. The dynamic power is decided by the input signal characteristics. When we apply a 100 mV amplitude spike signal as the input, the corresponding dynamic energy consumed per spike is about 0.077 nJ. Then, we can calculate the power consumption for one channel as:

$$P_t(nW) = 29.4 + 0.077 \cdot f, \quad (3.6)$$

where f is the firing rate of the spike signal. The total power dissipation for multi-channel SPD is given by:

$$P_{total}(nW) = 322 + 29.4 \cdot n + 0.077 \cdot f \cdot n, \quad (3.7)$$

n is the total number of recording channels.

Table 3.2: Power dissipation in biasing block.

Block	Power dissipation (nW)
Main Biasing	322
OTA	29.4
Current Squaring Circuit	0.7
LPFs	1.4

Figure 3.16: Layout and die photo of the prototype fabricated in 65 nm CMOS.

3.4 Measurement Results

To validate the function of the proposed design, we have implemented the system in a 65 nm CMOS process. The spike detector occupies a die area of $200\ \mu\text{m} \times 150\ \mu\text{m}$ as shown in Fig. 3.16. The input signals for measurement are constructed by selecting three distinct spike shapes from a database of 594 different spikes from intra-cortical neural recordings [30] described earlier in section 3.2.3. As mentioned earlier, the reason for choosing this database is several recent publications report their performance on this and it forms the basis of a fair comparison of performance. The signal-to-noise ratio can be defined as:

$$SNR = 20 \cdot \log \left(\frac{\sigma_x}{\sigma_n} \right), \quad (3.8)$$

where σ_x and σ_n are standard deviations of the signal and noise. Datasets with different SNR ranging from 15.5 dB to -2.6 dB are created by varying σ_n . To evaluate the performance of our

Figure 3.17: Experimental setup for the measurement of proposed SPD.

spike detector, we use false positive rate (FPR) and false negative rate (FNR) for comparison, which are defined following [6] [53] [23] as:

$$FPR = \frac{\text{number of false alarms}}{\text{number of total detection}} \quad (3.9)$$

and

$$FNR = \frac{\text{number of missed spikes}}{\text{number of actual spikes}}. \quad (3.10)$$

The experimental setup for the measurement is shown in Fig. 3.17. The input signals are fed to the IC through an arbitrary waveform generator (Tektronix AFG3022C) while the outputs are sampled using an FPGA. Figure 3.18(a) shows a sample of the input spike signal presented to the chip while Fig. 3.18(b) is the corresponding detection output for the SPD IC. As mentioned earlier, the chip can only be operated in mode I since the automatic threshold setting circuit has not been implemented in this version. So the ratio of threshold to signal amplitude is tuned externally for best performance while f_1 and f_2 are set to 400 Hz and 64 Hz respectively. It can be seen that the SPD is able to detect the spikes correctly amidst background noise. The SPD also functions correctly in the presence of large LFIN as shown in Fig. 3.19. In this case, 50

Figure 3.18: (a) Segment of the input simulated signal. (b) Measured output pulses of the chip showing time of detection.

mV spike signals are superimposed on top of a 10 Hz sinusoid mimicking low frequency LFP. Though a pure sinusoid is not a perfect approximation to LFP, we have tested the system with a wide range of frequencies of the sine wave to make sure the system functions well for a broad spectrum of input. It can be seen that the detection of the spikes is not affected even when the peak-to-peak amplitude of the sinusoid is increased from 50 mV in Fig. 3.19(a) to 400 mV in Fig. 3.19(b). Keeping the amplitude fixed at 400 mV, we have also increased the frequency of the interference and found no change in detection accuracy till the frequency increases beyond 70 Hz.

Next, to quantify the performance of the IC and compare it with other reported SPD [23]

Figure 3.19: (a) Original spike signal that is superimposed on low frequency interference. The SPD can correctly detect spikes even when the amplitude of low frequency interference is (b) same as or (c) 8X larger than spike amplitude.

[53] [56] and software simulated NEO, we show the performance of our spike detector with different noise levels in Fig. 3.20. The values on the X-axis of the plot are σ_n normalized to the spike amplitude. From these figures, we can note that the measured error rate of our design is only slightly higher than both simulations of our method and NEO. Moreover, it provides better detection accuracy compared to other recently reported SPD implementations.

Finally, we analyze the performance of the automatic threshold based detection in mode II. In this case, we report results from a MATLAB implementation of the algorithm proposed in Section 3.2.3. The averaging duration is set to $T_{avg} = 8$ ms. Figure 3.21(a) and (b) show the false negative and false positive rates in mode II when $f_2 = 64$ Hz and compare it with mode I (where threshold is fine tuned) for same value of f_2 . We also compare the results of automatic

Figure 3.20: Measured performance of the spike detection in mode I compared with other reported techniques. (a) Probability of missing detection and (b) Probability of false alarm with different noise level.

threshold setting with a more conventional choice of $f_2 = 2$ Hz in which case I_{LPF2} can be directly taken as background estimate and $I_{thr} = k \times I_{LPF2}$. It can be seen that the FNR for mode II is comparable to mode I while FPR is slightly worse at high noise. However, compared to automatic detection with conventionally chosen threshold, our method gives much better detection performance especially in high noise scenario.

The performance of the proposed method is even better compared to conventional setting of $f_2 = 2$ Hz when considering spikes with different amplitudes as shown in Fig. 3.22. In this case, the three spike waveforms were chosen to have different amplitudes with the maximum amplitude still same as all of the last cases. It was found in this case that the number of missed

Figure 3.21: Simulated performance of the spike detection in mode II compared with mode I and conventional automatic threshold setting. (a) Probability of missing detection and (b) Probability of false alarm with different noise level.

spikes increases significantly for the conventional case of $f_2 = 2$ Hz since the threshold estimate takes a long time to return to baseline. Figure 3.22(a) shows a time domain simulation of the above case (with lowest σ_n value) which depicts how our proposed threshold setting method in mode II with $f_2 = 64$ Hz is able to better track the true background level and not miss small spikes. Figure 3.22(b) depicts the same simulation for high σ_n —as expected from Fig. 3.13, there is not much difference between proposed and conventional threshold estimates in this case. The performance improvement due to the proposed method is quantified by comparing the FNR for the two cases while varying the ratio of minimum and maximum spike amplitudes as a parameter for the lowest σ_n value. The results depicted in Fig. 3.22(c) show that the

Figure 3.23: Possible circuit implementation of the algorithm for automatic threshold setting described in Fig. 3.14.

3.5 Automatic Threshold Setting

The SNR for the input signal will not vary very quickly. Instead, it will change slowly over time. However, for real-time neural recording, it is too difficult for an user to tune the threshold to keep the optimal performance over time. Hence, with this automatic threshold setting, people will not need to keep observing and calculating the SNR of the input neural signal. However, this can be done even with a slowly varying threshold without using a rapidly varying one as in our case. The major benefit of the rapidly varying threshold is that it can now settle to the true background estimate quickly after a spike making it less affected by the occurrence of neural spikes. Here, we describe a possible circuit implementation of the algorithm for automatic

Table 3.3: Table comparing this design with the state-of-the-art.

	Process (nm)	Area (mm^2)	VDD (V)	Power (μW)	Result	Feature	Style
CICC 2008 [6]	130	0.044	1.0	0.95	Test	NEO	Analog
ISCAS 2013 [24]	180	0.03	1.8	1.5	Simulation	NEO	Analog
ISCAS 2013 [25]	130	-	0.5	0.26	Simulation	fNEO	Analog
	130	0.05	0.5	0.13	Simulation	ED	Analog
TBCAS 2012 [23]	130	0.16	1.2	2.8	Test	Threshold	Analog
EMBC 2013 [57]	65	0.017	-	1.83	Simulation	NEO	Digital
EMBC 2013 [26]	130	6.17	-	85	Test	EC-PC	Digital
This work	65	0.03	0.7	0.04	Test	ED+LPF	Analog

threshold setting as shown in Fig. 3.14. In the circuit implementation shown in Fig. 3.23, a current controlled oscillator (CCO) and a counter together digitize and average I_{LPF2} . The CCO operates by charging a capacitor C_{osc} ($= 0.2$ pF) with the input current to create the voltage V_1 at the input of a hysteretic inverter. The moment V_1 crosses the upper threshold, all the inverters switch and the discharging process is initiated by turning on transistor M_{fb} . If reset current is much larger than input current, the CCO frequency can be shown to be proportional to the input current. After every T_{avg} seconds, the value of the counter is stored in a register as $\overline{I_{LPF2}}$ and the counter is reset. Depending on the value of flag that records whether a spike is detected in this averaging interval, I_{avg} and D_{th} registers are updated or not. The digital word D_{th} can be converted to I_{thr} using a compact current splitter based digital-analog converter (CSDAC) [51]. We have not implemented this circuit in the prototype IC described later; still we believe it is useful to show a possible compact implementation of the proposed algorithm to prove it can be integrated in a SPD block in future.

3.6 Detection Probability Estimator

Fig. 3.24 illustrates details of the current-controlled oscillator (CCO) based DPE. The capacitor C_{osc} is charged by the current $I_6 - I_7 - I_{thr}$. When V_1 crosses the higher threshold of the Schmitt Trigger inverter [58], it would pull V_2 down to the ground and V_o to VDD immediately. Then the feedback transistor M_{fb} turns on, discharging the capacitor C_{osc} . When V_1 reaches

Figure 3.24: Schematic of the oscillator based DPE circuit.

the lower threshold of the Schmitt Trigger inverter, the inverters will switch and C_{osc} will be charged by the currents again. Denoting the hysteresis window of the Schmitt Trigger inverter by ΔV , the frequency of this current-controlled oscillator can be obtained as:

$$f_{osc} = \frac{1}{\frac{\Delta V C_{osc}}{I_6 - I_7 - I_{thr}} + \frac{\Delta V C_{osc}}{I_{dis} - (I_6 - I_7 - I_{thr})}} \approx \frac{I_6 - I_7 - I_{thr}}{\Delta V C_{osc}} \quad (3.11)$$

since I_{dis} , the discharge current through M_{fb} , is very large. The CCO pulses clock a counter to create a digital representation of $I_6 - I_7 - I_{thr}$ which in turn is an estimate of the probability of detection. The counter is enabled for 1 ms every time there is a spike detection and the 6bit count, C_i for every consecutive $100\mu s$ is stored (total of 10 such values for 1 ms). We expect spike signals to have large amplitude and temporal duration exceeding 1 ms—noise signals that falsely trigger the detector will probably not satisfy at least one of the above criteria. To capture these two qualities, we use the following two features (f_1 and f_2):

$$f_1 = \sum_i C_i, f_2 = \max(C_i) \quad (3.12)$$

We have found that there is a lot of variability in C_1 . So the above features are computed excluding it. In the current chip, feature calculation is not included. Without it, the data rate for one channel for 50 detections/sec is 3kbps—a compression of $\approx 7X$. With feature extraction

Figure 3.25: Input neural signal, detection and oscillation outputs.

included on-chip, compression ratio would increase to about $25X$.

Figure 3.25 shows an example of one piece of input neural signal and the corresponding detection output and the CCO output before counting. The big advantage of the DPE is the weak dependence of performance the input amplitude or the threshold setting. To validate this, we test the DPE performance by varying the input signal level for a fixed value of threshold. Fig. 3.26, 3.27, 3.28 and 3.29 show the measurement results of the DPE for 2 different noise levels: 0.1 and 0.25. These numbers indicate the noise standard deviation relative to the amplitude of the spike classes—corresponding SNR values are 7 dB and 1.5 dB respectively. As the noise level increases, input spike signal is corrupted by the background noise more significantly, making it more difficult to separate the noise from spike signal. However, for the DPE we expect the values of f_1 and f_2 to increase equally for both signal and noise classes—thus separation between the two should not be affected.

Figure 3.26 and 3.28 show the distribution of features f_1 and f_2 for the two noise cases demonstrating clear separability of clusters of signal and noise. As expected, valid spike signals have large values for both f_1 and f_2 . Figure 3.27(a) and (b) demonstrate FPR and FNR when input signal strength is varied from 60 mV to 100 mV for same threshold level in the low

Figure 3.26: The DPE output feature distributions for low noise level.

noise scenario. The X axis plots the ratio of $I_{in,max}/I_{thr}$ normalized to the smallest value in each case. larger values of this ratio result in increased FPR for the conventional detection as can be expected if the threshold is too low compared to noise. However, for the DPE based detection, the clear separation between spike and noise features enable a drastic reduction of FPR using simple k-means clustering. A similar conclusion also holds for the high noise case in Fig. 3.29(a) and (b), though in this case FNR increases slightly for DPE.

3.7 Summary

We have reported a low power, low voltage, analog neural spike detector (SPD) for real-time neural recording system. The SPD saves power by approximating NEO using simple circuits and also uses current mode design enabling lowering of supply voltage. We also show that an integrated low pass filter with cut-off frequency $\approx 50 - 70$ Hz can be used to either track and reject low frequency interference from LFP (mode I) or provide a firing rate independent estimate of background noise for automatic threshold setting in SPD (mode II). Further, it uses a detection probability estimate to make performance relatively insensitive to threshold setting. We presented detailed simulations and measurements showing that the performance of our de-

(a)

(b)

Figure 3.27: The FNR and FPR plots for low noise level with DPE. .

sign is comparable with recent publications and software simulations of traditional NEO. Fig. 3.4 specifies the current status of each block. The SPD and DPE has been fabricated in a 65 nm CMOS process and the SPD occupies a die area of $200 \mu\text{m} \times 150 \mu\text{m}$. A significant portion of the chip is occupied by the capacitors. We made the input capacitance to be large enough (4 pF) to make sure the input current to be large enough for processing. This input capacitor C1 occupies an area of $100 \mu\text{m} \times 27 \mu\text{m}$. We also have a biasing block ($180 \mu\text{m} \times 50 \mu\text{m}$) which can be share by many other modules on the chip as well. Beside the spike detection, the feature extraction circuit is also included in this chip, which occupies around $40 \mu\text{m} \times 60 \mu\text{m}$

Figure 3.28: The DPE output feature distributions for high noise level.

Table 3.4: Current Status for Each Block

Block	Status
Spike Detector	Circuit Implemented
Automatic Threshold Setting	Matlab Simulated
DPE	Circuit Implemented

area. The digital implementation of the core function might be smaller—however, to implement a median threshold scheme, it needs to store a lot of values in memory which consumes a lot of area. Operating from a 0.7V supply, the SPD dissipates only 30 nW of static power per channel and 0.07 nJ of dynamic energy per input spike making it the lowest power consuming SPD reported till date. Simulations has been done to verify the function of automatic threshold setting in Matlab. In the next version of the chip, we will integrate the SPD, DPE and automatic threshold setting scheme into a single neural recording system.

For our proposed spike detector, we consider the average length of spike waveform to be 2.5 ms. All the parameter of the circuit, especially the corner of the two low pass filter is optimized based this value. SPICE simulations have been done to evaluate the capability for this circuit to detect a spike waveform with varying length. It will be easier for this circuit to detect a spike if the length is shorter. That means the potential changes a lot in a shorter time. After

(a)

(b)

Figure 3.29: The FNR and FPR plots for high noise level with DPE. .

derivative, there will be more power for the spike with a shorter length. On the other hand, if the length of the spike is longer, it means the input voltage changes very slowly with time. It might cause some difficulty for the circuit to detect a spike since energy after derivative calculation is reduced. It can be corrupted by the noise more easily. From simulations, at most a length of 10 ms spike signal can be detected by this circuit with a single output digital pulse. If the length of the spike waveform is ranging from 10 ms to 30 ms, one single input spike waveform will generate multiple digital pulses. If the duration of the spike waveform exceeds 30 ms, this circuit cannot distinguish it from noise.

Fig. 3.20 shows the measurement result to be slightly worse than simulation. There may be several reasons which caused this difference between simulation and measurement results. In simulations, discrete signal is utilized in the Matlab calculations. The derivative is calculated by the differences between two adjacent points. However in hardware implementation, the input is a continuous time signal while the derivative is achieved by a capacitor based differentiator which transfers the input voltage signal into a current. In Matlab simulation, we can quite easily achieve a perfect squaring function while in hardware implementation, current squaring implemented by the trans-linear circuit is also limited by a number of factors such as the body effect of MOSFET transistors, process variations and so on. The translinear loop in both the square block and filters depend on sub-threshold operation—hence, transistors operating near moderate threshold will cause a deviation from intended functionality. For the low pass filter, gain and corner frequency might also vary from mathematical simulation. Other factors like circuit noise and parasitic capacitor will affect the performance of fabricated circuit in some amount as well.

Chapter 4

Spike based Feature Extraction for Combined Spike Sorting and Intention Decoding

4.1 Background

As mentioned in previous chapters, action potentials contain the information encoded in the brain. For most scientific and prosthetic applications, relevant spike timing information for every neuron is required for neural signal decoding. Fig. 4.1(a) shows the system diagram of a typical neural recording system. The neural signal will be sensed and amplified by the analog front-end (AFE). Spike detection is to detect the neural spike from the background noise, which have been discussed in Chapter 3. Once a spike is detected, features will be extracted for spike sorting, which can be used to cluster the recorded neural spikes to the individual neurons. It lays the foundation for the intention decoding. The basic procedure for spike classification is generally as follows: (1) spike detection, (2) extraction of distinctive features from different spike shapes, and (3) classification of the spikes according to these features. Spike sorting methods are typically based on clustering predefined spike shape features such as peak-to-peak amplitude, width, or principle components [27] [28]. Several other analytical methods have been developed for spike sorting, for example: principal components analysis (PCA) [59], su-

perparamagnetic clustering [30], discrete wavelet transform (DWT) [60], discrete derivatives (DD) [34], template matching (TM) [61, 62]. etc..

Figure 4.1: (a) The flowchart for the conventional neural recording system (b) The flowchart for the proposed neural recording system.

4.2 Objectives

In this work, we propose a pulse-based feature extraction method using derivatives of the signal. It can be used in traditional spike sorting (by treating the number of pulses in a time window as a feature) or to enable a novel system architecture (Fig. 4.2(b)) where we use only one machine learner (ML). Typically, after the spike sorting another machine learner is needed (Fig. 4.2(a)) for neural intention decoding. The benefit of pulse-based feature extraction is that the features (spatio-temporal pulse train) can be directly utilized for neural signal decoding (Fig. 4.2(b)). [63] states only spike detection output is often good enough for intention decoding thus allowing the use of one ML. However, spike detection output is just pulse train, with pulses appearing whenever there is a spike independent of the waveform of the spike. Encoding some information about spike shape in the pulse train makes it richer than just spike detection and will result in higher decoding accuracy.

Figure 4.2: Block diagram of the feature extractor. The differentiator and the current rectifier can be shared with the spike detector block described in Chapter 3.

Here, we describe such a pulse-based feature extractor circuit and present measurement results from a prototype fabricated in 65nm CMOS. We do not show complete results of intention decoding here—instead, we show as a first step that pulse-based features contain sufficient information for spike sorting. K-means clustering is utilized for unsupervised classification of the pulse-count features. In future, an ML such as the one in [64] will be used to perform the decoding directly.

4.3 System Architecture

The architecture of the feature extractor is shown in Fig. 4.2. The differentiator and the current rectifier is same with the current-mode spike detector we proposed in [65]. These block can be shared for the combined detection and sorting system. The differentiator will get the derivative of the input signal, which will be rectified later by the current rectifier. Both the positive part and the negative part will go to a CCO. The CCOs (CCO1 and CCO2) generate digital pulses V_{out1} and V_{out2} which will be counted by the counters CNT1 and CNT2 followed by which results F_1 and F_2 are to be saved as the features.

Figure 4.3: Circuit diagram of the current rectifier that uses a feedback amplifier (FA) to pin the input node and (b) detailed circuit of the feedback amplifier (FA).

Figure 4.4: Detailed circuit of the feedback amplifier (FA).

4.3.1 Differentiator

A capacitor C_{in} ($= 4$ pF) is used as a differentiator which will convert the input voltage signal into a current proportional to (dV_{in}/dt) . Larger values of C_{in} will produce more input current at the expense of chip area. The value of 4 pF is chose to ensure at least ≈ 1 nA of current is created for a 50 mV amplitude spike signal.

Figure 4.5: The schematic of the CCO.

4.3.2 Current Rectifier

Since the current squaring circuit can only process uni-directional current, a class-B current mirror similar to [50] and shown in Fig. 4.3 is adopted as a rectifier before it. A negative feedback is added in the class-B current mirror through an OTA (with a bias current $I_a = 20$ nA) which reduced the input impedance by a factor equal to its open-loop gain. It also ensures that one terminal of the capacitor stays at a fixed voltage enabling it to act as a differentiator. This feedback amplifier (FA) employs long channel length transistors instead of cascodes to get high gain at low voltages. The bias current is chosen to have a bandwidth of ≈ 20 kHz that is several times larger than the ≈ 5 kHz bandwidth of a typical spike. A class-B current mirror has a deadzone that limits the minimum detectable input current [50]. This deadzone is directly proportional to parasitic capacitance C_{gs} of transistors M_n and M_p . To reduce this, both of these transistors are kept as minimum sized core transistors. Further, these transistors have low V_T to enable reduction of power supply. The simulated value of the minimum detectable signal at V_{in} due to this deadzone is approximately 100μ V.

Figure 4.6: The oscillation waveform of the CCO.

4.3.3 Current-Controlled Oscillator (CCO)

Fig. 4.5 illustrates detail circuit of the current-controlled oscillator *CCO2*. *CCO1* is the opposite type of *CCO2*, i.e. it has the PMOS and NMOS transistors swapped. The CCO we are using for feature extraction has a same structure with the one employed in DPE in Chapter 3. However, for the consistency of description and the convenience of reading, we present the details of the schematic here again. The capacitor C_{osc} is charged by the current $I_2 - I_{thr}$, where I_2 is generated from the current rectifier. When V_1 crosses the higher threshold of the Schmitt Trigger inverter, it would pull V_2 down to the ground and V_{out} to VDD immediately. Then the feedback transistor M_{fb} turns on, discharging the capacitor C_{osc} . When V_1 reaches the lower threshold of the Schmitt Trigger inverter, the inverters will switch and C_{osc} will be charged by the currents again. Denoting the hysteresis window of the Schmitt Trigger inverter by ΔV , the

Figure 4.7: The three template spike shapes with a noise level of 0.05 and the corresponding output waveforms of the two CCOs.

frequency of this current-controlled oscillator can be obtained as:

$$f_{osc} = \frac{1}{\frac{\Delta VC_{osc}}{I_2 - I_{thr}} + \frac{\Delta VC_{osc}}{I_{dis} - (I_2 - I_{thr})}} \approx \frac{I_2 - I_{thr}}{\Delta VC_{osc}} \quad (4.1)$$

since I_{dis} , the discharge current through M_{fb} , is very large. The oscillation waveform of the CCO is illustrated in Fig. 4.6. The CCO pulses clock a counter to create a digital representation of $I_2 - I_{thr}$ which in turn will generate the features for the spike signal. The counter is enabled for 3 ms every time there is a spike detection and the 6 bit count, F_1 or F_2 for every consecutive $50\mu s$ is stored (total of 80 such values for 4 ms).

4.4 Algorithm: pFSDE Features

For simulations, the spike signals are chosen from the database developed in [30] that is commonly used in many recent publications on spike detection circuits [24] [54] and hence provides a baseline for comparison. It comprises 594 different average spike shapes compiled from

recordings in the neocortex and basal ganglia. Spikes randomly selected from the database were superimposed at random times and amplitudes aiming to mimic the background noise of actual recordings generated by distant neuron-activity. A train of spikes having 3 distinct shapes (chosen from the same database), at a desired frequency, is then superimposed on the background; spike-amplitude is normalized to 1. The background noise level is determined by its standard deviation with respect to the spike-amplitude, ranging from 0.05 to 0.2 corresponding to SNR in the range of 3.5dB–15.5dB.

Figure 4.6 shows three different spike shapes from one of the datasets with a background noise level of 0.05. SPICE simulations of the outputs of the 2 CCOs in a 65nm-CMOS process are also illustrated in this figure to give an intuition about the nature of pulse trains produced. Average duration of a spike in the waveform is about 3 ms. As mentioned in Section 4.3.3, each counter gives pulse-counts C_i in a time-window T_{int} and is then reset to start count again from 0. The n^{th} digital-pulse-count for the i^{th} CCO is then given by:

$$C_{i,n} = \int_{(n-1)T_{int}}^{nT_{int}} f_{osc,i} dt \quad (4.2)$$

where $i = 1, 2$ and $f_{osc,i}$ is given by Eq. (4.1). From each counter, $N = T_{en}/T_{int}$ values are obtained for an individual spike, where $T_{en} = 4$ ms is the time for which counts are considered every time there is a detection. Thus, $n = 1, 2, \dots, N$ and we have $N = 80$ such counts for $T_{int} = 50\mu s$. For the 2 CCOs, a total of 160 values represent the features of the spike shape in the 4-ms window of observation. To reduce dimensionality, only derivative extrema features inspired by [66] were used. It follows that the highest of the counts from the first/second CCO estimates the maximum/minimum of the first derivative of the spike signal where the derivative is simply the output current from the differentiator. The second derivative of the spike signal can be estimated by the differences between consecutive pulse-counts from each CCO except at the zero crossing where the derivative signal shifts from one channel to the other. We could have combined the pulse count differences from both channels at the zero-crossing to get a good estimate of second derivative; however, since our motive is to just use these estimates for classification, we chose

Figure 4.8: Input neural signal and the outputs coming from CCO1 and CCO2.

to simply take the highest of the pulse count differences from both channels *separately* keeping in mind that this might not reflect the true maxima of second derivative. Hence, finally, we have a dimensionality $m = 4$ considering both channels. In summary, the features required for this method are given by:

1. $F_1 = \max \{C_{1,n}\}$
2. $F_2 = \max \{C_{2,n}\}$
3. $F_3 = \max \{C_{1,n} - C_{1,(n-1)}\}$
4. $F_4 = \max \{C_{2,n} - C_{2,(n-1)}\}$

for $n = 1, 2, \dots, N$. A mathematical model built in MATLAB (with $N = 80$) extracts the features by our proposed method, which from now on in the text will be referred to as pulse-based First and Second Derivative Extrema (pFSDE).

Figure 4.9: Comparison of PCA (for 3 and 10 PC's) with pFSDE performance in spike clustering. First we can see that FSDE works better for white noise than the noise added in the dataset comprising similar spike shapes. Second, simulations and chip measurements show FSDE and pFSDE works better than PCA for high noise conditions.

4.5 Results

We have performed spike-clustering by the pFSDE for various background noise levels and compared it with the conventional PCA-based spike sorting method. PCA has been done on the raw spike waveform and clustering is performed on the top $m = 3$ or $m = 10$ principal components separately.

4.5.1 Simulation

In MATLAB, detected spikes from one of the datasets are simply fed to the feature extraction algorithm and the output pulses every $T_{int} = 50\mu s$ are counted (giving 80 pulse-counts for each spike shape), followed by the extraction of the count-values representing the first and the second derivative-extrema. The original sampling rate for the input data is 24 KHz, quite low for simulation. So the input data is first interpolated, followed by PCA on the up-sampled spike signal. After application of the K-means clustering algorithm on the PCA-features ($m = 3$

Figure 4.10: Variation of classification error for pFSDE with pulse-count per spike for varying background noise levels. Note optimal pulse-count (corresponding to a specific threshold) for each curve in the plot.

features or PC's for PC_3 and $m = 10$ for PC_{10}) as well as the extracted features by pFSDE, the misclassification error in each case has been plotted against background noise level (given by normalized standard deviation σ_n) in Fig. 4.9. In Fig. 4.9, simulation results for spike sorting with white noise as the background are also provided. The performance is even better if the background noise has Gaussian distribution.

It suggests that as the background noise increases, the error rises for both pFSDE and PCA. As expected, PC_{10} itself has a more accurate performance than PC_3 , due to greater raw spike-signal information in the greater number of features. Notably, the accuracy of our proposed feature extraction method is comparable with that of the conventional PCA-based feature extraction for low noise, and performs substantially better than the latter for higher noise levels. Moreover, the proposed feature extractor has a much simpler structure and some of the sub-blocks can be shared together with the spike detection. In addition, the extracted pulse-based features can be directly used for intention decoding.

Figure 4.11: Microphotograph of the feature-extractor IC.

Next, the accuracy of the pFSDE method was tested by varying the quantization level of the counts that depends on the pulse-rate (PR) of the CCO. Higher PR can be obtained for lower threshold of the inverter in the CCO and leads to less quantization noise. A MATLAB implementation was done to explore possible trade-offs between accuracy and CCO PR for the pFSDE method. From the results depicted in Fig. 4.10, we see that with increasing noise level, accuracy degrades with increased PR. Intuitively, when the input signal is itself noisy, higher PR just quantizes the noise and leads to poor result. Hence, there exists an optimal PR depending on noise level which implies future designs should incorporate mechanisms for setting PR based on noise levels.

4.5.2 Measurement

An IC with the feature extraction circuits has been fabricated in 65nm CMOS (Fig. 5.22). For testing of spike-sorting accuracy, neural signals from the same dataset [30] were used. Fig. 4.12 shows a piece of neural signal as the input and the measured CCO outputs V_{out1} and V_{out2} . The counter circuits were implemented on an FPGA off-chip. The parameters, again, were $T_{en} = 4\text{ms}$ and $T_{int} = 50\mu\text{s}$ so that $N \approx 80$ counts were available for each spike shape. The 4 pFSDE features were extracted from the counts as described earlier. Misclassification error from application of K-means on the chip data has been plotted against noise in Fig. 4.9. Chip

Figure 4.12: Input neural signal and the measured outputs coming from CCO1 and CCO2.

results are found to be slightly worse than those from software, but still comparable with PCA for low background noise and considerably better for noisier signals.

During measurement, a pico-ammeter (Keithley 6485) measured the average current from the power supply (V_{dd}) to estimate the system power dissipation, which is mostly dependent on input spike frequency f_{in} . Results show that power dissipation roughly varies as f_{in} . Supplied from a V_{dd} of 1V, total power dissipation is measured to be roughly $2\mu\text{W}$ and $3.4\mu\text{W}$ for $f_{in} = 100\text{Hz}$ and 200Hz respectively. The performance of the chip is summarized and compared with other relevant work in TABLE 4.1.

Table 4.1: Comparison Table

	[67]	[9] ¹	This work ²
Technology	$0.18\mu\text{m}$	90nm	65nm
Area (mm^2)	0.178	0.06	0.03
Supply voltage (V)	1.8	0.55	1
Power dissipation (μW)	55	2.03	2 ³
Method	Analog	Digital	Analog

¹ Including spike detection, alignment and feature extraction.

² Including spike detection and feature extraction.

³ Input spike firing rate is 100 Hz.

4.6 Summary

We presented a pulse-based feature extraction method using first and second derivative maxima of neural signals. The VLSI circuit is designed in 65nm CMOS and dissipates $2\mu\text{W}$ of power from $V_{dd} = 1\text{V}$ for an input spike rate of 100 Hz. Preliminary results of classifying noisy spikes (from a commonly used spike-waveform dataset) using features extracted by the chip show that the accuracy is comparable with PCA results for low noise and substantially better ($\approx 10.5\%$) for noisier signals. In future works, we will feed these features directly into an ML for intention decoding [64].

Chapter 5

VLSI Extreme Learning Machine: A Design Space Exploration

As mentioned in the previous sections, classification is a very important block in the smart sensor system. Inspired by the efficiency and robustness of human brain, neuromorphic machine learning system provides low power, real-time operation and has a wide range of applications. In this work, we propose to use a machine learning co-processor (MLCP) along with a conventional digital processor in a smart sensor. Using a MLCP has several advantages over using a conventional processor for machine learning: During the training period, machine learner based classifier could be able to “learn” and adapt to different application environment; It has a better fault tolerance; and it is more capable of processing high dimensional data with lower energy per operation. This chapter proposed a novel hardware implementation of the extreme learning machine (ELM) algorithm. This classifier can be used in the neural recording system for spike sorting and neural signal decoding. However, this ELM classifier can also be applied in the smart sensors for other applications.

5.1 Background

Historically, analog computing has been difficult to scale to large systems due to several reasons, a major one being device mismatch [68]. The effect of mismatch on traditional circuits like differential amplifiers and current mirrors is well documented [69]. It has also been

Table 5.1: Comparison of Different Methods for Hardware Machine Learner.

Method	Advantages	Disadvantages	Reference
Digital	High speed	Digital memory: High power dissipation; Large area	[10, 35, 38]
Analog	Low power consumption	Floating-gate transistors requirement: High voltage; Analog Interface	[11, 47]
Mixed-Signal	No memory and floating-gate transistors needed	Multiplying DACs consume large area	[78]

shown that for MOS based circuits, the extra power dissipation needed to overcome effects of mismatch can be an order of magnitude higher than the limit imposed by thermal noise [68]. With transistor dimensions reducing over the years, variance in properties of transistors, notably the threshold voltage, has kept on increasing making it difficult to rely on conventional simulations ignoring statistical variations. The problem is particularly exacerbated for neuro-morphic designs [70], where transistors are typically biased in the sub-threshold region [71–73] of operation (to glean maximal efficiencies in energy per operation) since device currents are exponentially related to threshold voltages thus amplifying its variations as well. For example, it is shown in [74] that an array of 5 – *bit* DACs in $0.35\mu\text{m}$ CMOS process used as tunable weights only provide an effective of 1.1 bits due to mismatch. In general, there has been an approach to compensate for mismatch either through floating-gates [75] or otherwise [70]. Digital calibration can be used to compensate for these effects [74]. However, they lead to huge area overheads due to the requirement of extra transistors for calibration and storage of digital bits [76]. Sometimes, it is claimed that learning can compensate for mismatch and has been demonstrated in specific cases [?, 77]—but the claim needs to be further quantified using standard datasets since mismatch will exist in the learning circuits as well. A table comparing different methods for the implementation of hardware machine learner is illustrated in Table 5.1.

5.2 Objectives

The goal of this work is to develop a machine learning circuit that has the following features:

- (1) Has lower energy requirements compared to conventional digital implementations.

- (2) Requires small area and hence can be scaled to multi-core implementations.
- (3) Does not need high voltages and is compatible with conventional CMOS technologies.
- (4) Implements an algorithm that can be trained quickly and efficiently.

The ELM algorithm is popular in the machine learning community due to its fast training speed and has been shown to produce similar or better performance compared to support vector machines (SVM) [48]. A closely related method (termed Neural Engineering Framework) has also been used to generate large scale models of cognitive systems [79]. ELM based methods have been used to classify spike time based patterns recently [80] and online learning algorithms for ELM have been proposed [81]. Clearly, a hardware implementation of this algorithm would satisfy objective 4 listed above. In this chapter we present a circuit that ‘utilizes’ mismatch in analog circuits to do effective computation in the first layer of a two layer spiking neural network implementation of ELM. This approach can be used in other algorithms like liquid state machine (LSM) or echo state networks (ESN) since they require random projections of the input as well. Using analog approaches helps us fulfill objectives (1) and (2), while not using floating-gates makes this approach free from high-voltage requirements as in objective (3). We have earlier proposed the idea of using spiking neurons for implementing ELM [82] and described the advantages of such an architecture over standard digital implementations [83]. It should be noted that this method only exploits spiking neurons for ease of hardware implementation and does not use any spike based learning rules to perform the learning of the second stage. The major hardware benefits are the use of low-power analog circuits for the reservoir and simple digital circuits for the second stage. We demonstrated the first VLSI implementation of this principle in [64] where it was used for decoding motor intentions for implantable brain-machine interfaces. In this chapter, we present a different chip utilizing the same core circuit as [64] but operating on 10 bit digital inputs instead of spikes. In future, we can apply this chip for epileptic seizure detection from EEG and arrhythmia detection from ECG. Instead of a specific application, this chapter presents an entire design space trade-off between speed, power and accuracy. Finally, we present a method and associated circuits to virtually expand the input

Figure 5.1: The architecture of ELM algorithm. d is the dimension of the input data and L denotes the number of hidden layer neurons.

and output dimensions of the chip beyond the physically implemented 128 channels. We show results of applying inputs from standard machine learning data bases such as [84].

In the next section, we present details of the ELM algorithm and training methods. Section 5.4 describes the VLSI architecture of the chip and details of the sub-circuits. The trade-offs between noise, speed and energy dissipation of this architecture are presented in Section 5.5. An important limitation of hardware machine learners is limited input and output dimensions. In section 5.6, we present a method to virtually expand the dimensions beyond the physical number of channels on the chip. Measurement results are presented in Section 5.7 and finally we conclude in the last section.

5.3 ELM Theory

In this section, we will present a brief description of the ELM algorithm and refer the readers to [48, 85] for details. As illustrated in Fig. 5.1, the ELM algorithm is applicable to a two layer neural feed-forward network with L hidden neurons having an activation function $g : R \rightarrow R$. Without loss of generality, we consider a scalar output in this case though the method can be

easily extended to multiple outputs by considering each output one by one [86]. The output of the network o is given by:

$$\begin{aligned} o &= \sum_i^L \beta_i H_i = \sum_i^L \beta_i g(z_i) \\ &= \sum_i^L \beta_i g(\mathbf{w}_i^T \mathbf{x} + b_i), \mathbf{w}_i, \mathbf{x} \in R^d, \beta_i, b_i \in R, \end{aligned} \quad (5.1)$$

where β denote the output weights, z_i and H_i are the input and output of the i -th hidden layer neuron. $\mathbf{w}_i$ denotes the input weight and b_i is the bias for the i -th neuron. In general, a sigmoidal form of $g()$ is assumed though other functions have also been used. Compared to traditional back propagation learning rule that modifies all the weights, the ELM allows $\mathbf{w}_i$ and b_i to be random numbers drawn from any continuous distribution while only the output weights, β_i needs to be tuned based on the training data T . For N samples $(\mathbf{x}_k, t_k)$, the hidden layer output matrix $\mathbf{H}$ is defined as:

$$H = \begin{bmatrix} g(\mathbf{w}_1^T \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_L^T \mathbf{x}_1 + b_L) \\ \vdots & \dots & \vdots \\ g(\mathbf{w}_1^T \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_L^T \mathbf{x}_N + b_L) \end{bmatrix} \quad (5.2)$$

The desired output weights, $\hat{\beta}$ are then the solution of the following optimization problem:

$$\text{Minimize}_{\beta} : \| \mathbf{H}\beta - \mathbf{T} \|^2, \quad (5.3)$$

where $\beta = [\beta_1 \dots \beta_L]$ and $T = [t_1 \dots t_N]$. The ELM algorithm proves that the optimal solution $\hat{\beta}$ is given by $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$ where $\mathbf{H}^\dagger$ denotes the Moore Penrose generalized inverse of a matrix [48]. The huge benefit of this method is that it removes the need for iterative tuning and gives a simple formula to calculate the weights. The orthogonal projection method can be efficiently used to find $\mathbf{H}^\dagger$ as $(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$ if $\mathbf{H}^T \mathbf{H}$ is non-singular or as $\mathbf{H}^T (\mathbf{H} \mathbf{H}^T)^{-1}$ if $\mathbf{H} \mathbf{H}^T$ is nonsingular. Further, using concepts from ridge regression theory [87], a small constant I/C is often added

Figure 5.2: System architecture of the mixed signal integrated circuit that implements the first stage of ELM; the second stage is implemented in digital domain. The digital input data is converted to current by the IGC and then multiplied by random weights w_{ij} in the current mirror array. The current is converted to digital domain by the combination of a spiking neuron and a counter.

to the diagonal of $\mathbf{H}^T \mathbf{H}$ or $\mathbf{H} \mathbf{H}^T$ of the Moore-Penrose generalized inverse $\mathbf{H}$ —the resultant resolution is stabler and tends to have better generalization performance. The value of C is typically optimized as a hyperparameter using cross-validation techniques.

5.4 System Architecture

The architecture of the proposed mixed signal classifier that exploits analog computing for the $d \times L$ random weights of the input layer is shown in Fig. 5.2. The corresponding timing diagram is shown in Fig. 5.3. The input data ($Data_in$) will be fed to the particular channel in the system serially through a 1 to 128 demultiplexor according to the corresponding address $A < 6 : 0 >$ through a serial peripheral interface (SPI). The number of bits (NOB) of $Data_in$ for each channel is $b_{in} = 10$. Input data will be stored in shift registers first for the configuration of the current-mode digital-to-analog convertor (DAC) in the input-generation-circuit (IGC). The

Figure 5.3: Timing diagram of the ELM system where RN_in is a global reset, $Data_in$, A and CLK_in are SPI control signals to transfer input data to the IC. NEU_EN enables the neuron to produce spikes while CLK_cnt is used to read out the counter values C one by one.

function of IGC is to generate an analog DC current according to the input data which will be copied to every column using a current mirror. Multiplied by the random weights generated in the current mirror array, the current in one column will be summed according to Kirchoff's current law (KCL) and flow into a hidden layer neuron. This current is denoted as I_i^z for the i -th neuron in Fig. 5.2 and is analogous to the variable z_i in Fig. 5.1. Spiking oscillations with different frequency will be generated by the neuron according to their own input currents which is counted by an asynchronous counter forming a row of the matrix $\mathbf{H}$. Through a column scanner, these hidden layer outputs can be transferred to the FPGA to first get the output weight β during training and later for the second stage computation of ELM during regular operation. Other timing and control signals will also be provided by the FPGA as shown in Fig. 5.3. Next, we describe the operation of each block.

5.4.1 Input Generation Circuit (IGC)

Figure 5.4 shows the schematic of the input generation circuit for each dimension of input. The reference block provides a fixed master biasing current I_{ref} that acts as the reference current of the current DAC as well as the biasing for the active current mirror. The input data $Data_in$ is applied to configure a $b_{in} = 10$ bits MOS based current splitting DAC to generate a

Figure 5.4: Schematic of input generation circuit (IGC) for one channel. A reference current is split according to the 10 bits of input data to create I_{DAC} . The capacitor C ensures sufficient SNR when the current is mirrored to the L columns. An active current mirror is enabled to allow fast settling when I_{DAC} is small.

corresponding analog current [51]. The output current of this DAC is given by:

$$I_{DAC} = (2^{-1}D_9 + 2^{-2}D_8 + \dots + 2^{-9}D_1 + 2^{-10}D_0) I_{ref}. \quad (5.4)$$

I_{DAC} is multiplied with the input weights by current mirroring operation as described later. A capacitor $C = 0.4\text{pF}$ is also added at the gate of the current mirror array for each row to improve noise performance—this will be discussed in the later section. In the conventional current mirror, bandwidth is in proportion to the input current. If $Data_in$ is too small, the settling time of the current mirror might be too large. To solve this problem, an active current mirror is added to complement the conventional current mirror. Switch S_1 is used to turn on the active current mirror if all of the 4 MSBs are zero. When all the bits of $Data_in$ are 0, switch S_2 turns on to pull V_{bias} to ground and shut off the current mirrors in that row. The logical signals to control

Figure 5.5: Schematic of the neuronal oscillator circuit followed by an asynchronous counter. The neuron is enabled when control signal NEU_EN is high. The capacitors can be digitally reconfigured and have the following values: $C_{a1} = 100\text{fF}$, $C_{a2} = 200\text{fF}$, $C_{b1} = 50\text{fF}$, $C_{b2} = 100\text{fF}$.

$S1$ and $S2$ are given by:

$$\begin{aligned} S_1 &= \overline{D_6 + D_7 + D_8 + D_9}. \\ S_2 &= \overline{D_0 + D_1 + \dots + D_8 + D_9}. \end{aligned} \quad (5.5)$$

where D_i are the bits of $Data_in$.

5.4.2 Neuron

Figure 5.5 details the circuit of the hidden layer neuron block. It is a current-controlled oscillator structure followed by an asynchronous counter. This is one of the simplest neuron circuits described in [88]. This circuit has the issue of large short-circuit current dissipation in the inverters. However, in our case we can avoid this problem by operating at very low power supply voltages ($\approx V_{TN} + V_{TP}$) making the short-circuit current negligible. The neuron is enabled when the control signal NEU_EN is high. The oscillation waveform at the nodes V_{mem} and V_{out} are illustrated in Fig. 5.6. V_{mem} is charged down by the input current $I^z - I_{lk}$ till

Figure 5.6: Oscillation waveforms at different nodes of the neuron circuit.

it reaches the threshold voltage of the inverters. At that point both the inverters trip making the output switch to ground. Since the voltage change at the node of V_{out} is VDD, the voltage change of V_{mem} due to the feedback capacitor is given by:

$$\Delta V_{mem} = \frac{C_b}{C_a + C_b} VDD. \quad (5.6)$$

Also, the reset transistor turns ON charging V_{mem} up by the current $I_{rst} + I_{lk} - I^z$. The inverters trip again once V_{mem} reaches the threshold and this process continues as long as NEU_EN is high. Both the capacitors C_a and C_b can be digitally reconfigured as shown in Fig. 5.5. The values of the capacitors are: $C_{a1} = 100\text{fF}$, $C_{a2} = 200\text{fF}$, $C_{b1} = 50\text{fF}$, $C_{b2} = 100\text{fF}$.

We can derive an equation for the oscillation period T_{sp} . It is composed of two parts: the time T_1 for the input current I^z to discharge the capacitor of node V_{mem} and the time T_2 to reset the capacitor. Hence, T_{sp} is given by:

$$T_{sp} = T_1 + T_2 = C_b VDD \left(\frac{1}{I^z - I_{lk}} + \frac{1}{I_{rst} - I^z + I_{lk}} \right). \quad (5.7)$$

Assuming $I_{lk} \approx 0$, the relationship between the neuron spiking frequency and the input current I^z can be easily obtained as:

$$f_{sp} = g(I^z) = \frac{I^z (I_{rst} - I^z)}{I_{rst} C_b VDD}. \quad (5.8)$$

Figure 5.7: (a) Neuron spiking frequency initially increases with the increase of the input current I^Z till $I^Z = I_{flx}$. It then reduces and becomes zero finally when $I^Z = I_{rst}$. (b) The transfer function (solid line) of the neuron with input I^Z and output H can be saturated at a pre-defined value of 2^b by stopping the counter.

This quadratic relationship of equation (5.8) between current and frequency is plotted in Fig. 5.7 (a). As we can see from Fig. 5.7(a), if $I^Z \ll I_{rst}/2$, we have almost a linear relation given by:

$$f_{sp} \approx \frac{I^Z}{C_b VDD} = K_{neu} I^Z, \quad (5.9)$$

$$K_{neu} = \frac{1}{C_b VDD}. \quad (5.10)$$

where $K_{neu} = \frac{1}{C_b VDD}$ denotes a conversion gain from current to frequency. When $I^Z = I_{rst}/2$, f_{sp} will reach its maximum value f_{max} . After this point, the spiking frequency will keep falling down till it reaches zero for $I^Z = I_{rst}$. Since the inflection point of the curve is reached at $I^Z = I_{rst}/2$, we refer to this current value as I_{flx} . The chip has digital control bits making the capacitors configurable. As shown in Fig. 5.5, an asynchronous counter counts the total number of spikes from the neuron during a fixed period of time T_{neu} (time duration for which

Figure 5.8: Comparison of neuron spiking frequency between theory and simulation in SPICE show close match.

NEU_EN is high) and generates the output H . A hard nonlinearity in the form of saturation can be implemented by stopping the counter whenever its count reaches a pre-defined limit 2^b . b in this case is the valid MSB of the counter output which is also configurable from 6 to 14. If only the linear region of the neuron spiking waveform is adopted (this is also the most energy efficient part as shown later), the final transfer function of the hidden layer neuron can be represented by:

$$H = \begin{cases} f_{sp} T_{neu} (\approx K_{neu} I^z T_{neu} \text{ if } I^z < I_{flx}), & \text{if } H < 2^b \\ 2^b. & \text{otherwise} \end{cases} \quad (5.11)$$

This saturating nonlinearity is shown in Fig. 5.7(b). This nonlinearity was preferred due to its ease of implementation and digital control. From Fig. 5.7(b) we can also note the current at which the H saturates is denoted by I_{sat}^z . This value depends on both T_{neu} and b . Also, $[0 \ I_{max}^z]$ is used to denote the range of input currents to the neuron.

Figure 5.8 plots SPICE simulation of the neuron spiking frequency with the variation of

Figure 5.9: Simulated neuron spiking frequency with increasing input current for 3 different VDD. The curves saturate at higher maximum frequencies for higher VDD. Note the logarithmic scales for both plots.

input current I^z on a logarithmic scale and compares it with theoretical predictions based on equation 5.8. For this simulation, C_a and C_b were set to be 300fF and 50fF respectively while VDD was kept at 1V. As expected, the spike frequency increases linearly for small values of I^z , reaches a maxima eventually and then starts reducing for further increase in I^z . Results from a similar simulation but for three different values of VDD (0.8, 1 and 1.2 V) are shown in Fig. 5.9. Since f_{sp} is inversely proportional to VDD, f_{sp} is higher for small I^z with a smaller VDD. However, when VDD is lower, I_{rst} is smaller and hence f_{sp} attains the peak value at smaller value of I^z , i.e. I_{flx} reduces when VDD is reduced. On the other hand, for higher VDD, f_{sp} saturates at a larger value f_{max} and it is attained for larger value of I_{flx} .

5.4.3 Current Mirror Array

The digital input $Data_{in}$ is mapped to a vector of input current $\mathbf{I}_{in}$ which are copied to every neuron using a current mirror. These inputs can also be obtained from a sensor such as a photo diode. The capacitor $C = 0.4\text{pF}$ is kept to maintain a minimum SNR [11] at the ex-

pense of bandwidth. For low-power operation, we operate the current mirrors in sub-threshold regime. Minimum sized transistors are employed in these current mirrors to exploit VLSI mismatch which is necessary for the generation of random input weights w_i and bias b_i of ELM. For example, the contribution of input $i_{in,i}$ to the total input current of neuron j is given by $i_{in,i}w_0e^{\Delta V_{T,ij}/U_T}$ where U_T is the thermal voltage, w_0 is the nominal current mirror gain while $\Delta V_{T,ij}$ denotes the mismatch of the threshold voltage for the transistor copying the i -th input current to the j -th neuron. This last term is a random variable with a Gaussian distribution and hence the weights $\mathbf{w}$ in equation (5.1) above get mapped to random variables with a *log-normal* distribution in our implementation. Since in our implementation $w_0 = 1$, we can write:

$$w_{ij} = e^{\frac{\Delta V_{T,ij}}{U_T}} \quad (5.12)$$

Do note that the ELM algorithm only requires random numbers from any continuous distribution [86]. Here, we choose log-normal distribution due to the intrinsic physics of sub-threshold mosfets. If biased in above-threshold regime, the distribution of random numbers would be closer to Gaussian.

Figure 5.10: Simplified circuit diagram of one current mirror for noise analysis.

5.4.4 Parameter Choice

To determine the performance of the network, we chose two representative tasks of regression ($d = 1$) and classification ($d = 14$). For the regression task, the network was given a set of noisy samples and had to approximate the underlying function. For classification, six different data sets with widely varying dimensions and training set sizes were chosen from the UCI machine learning repository [84]. Here, we show results for only the ‘brightdata’ case as a representative but the conclusions drawn are valid across the other data sets. It is a two class problem results show that mismatch in these that includes 1000 training data and 1462 testing data. The reasons for choosing these tasks were that the performance of the software implementation for these tasks are reported in publications as a typical benchmark [48].

For the following simulations done in MATLAB, we considered the mismatch in current mirror weights as the dominant factor. It was assumed to be log-normally distributed with a standard deviation of V_T , σ_{V_T} ranging from 5 to 45 mV (as a reference, σ_{V_T} in our fabricated chip is ≈ 16 mV). Equation (5.11) was used to simulate the neuronal characteristic and the other parameters were kept at fixed nominal values of $K_{neu} = 26\text{KHz/nA}$ and $T_{neu} = 56\mu\text{sec}$. In real applications, variations exist for other parameters in the neuron transfer function as well. However, simulation results show that mismatch in these do not affect the qualitative nature of the results we present here. Hence, for simplicity we keep them fixed in the simulations presented here.

5.4.4.1 Input Mapping

For efficient use of the hardware, we need to determine how to map the compact set $X = [-1, 1]$ to input currents. First, it can be only mapped to a set in R^+ since we have unidirectional current mirrors. Assume the maximum input current for one dimension is I_{max} , i.e., the set is $[0, I_{max}]$. Therefore the maximum current going to the neuron $I_{max}^z = d \times I_{max}$. From Fig. 5.7(b), we need to find out the relationship between I_{max}^z and I_{sat}^z . Though theoretically any positive set will work, it might need an unreasonably large number of neurons to get a satisfactory performance. To illustrate this point intuitively, consider a case where $I_{max}^z < I_{sat}^z$. Then the

Figure 5.11: Variations of L_{min} with I_{sat}^Z / I_{max}^Z show that the optimal value of this ratio is ≈ 0.75 .

transfer function of the neuron is a linear function without any high order components. Also, if $I_{max}^Z \gg I_{sat}^Z$, the outputs of most neurons will be saturated to 2^b , and will not encode the variations of the input. Both of these cases will require a large number of hidden layer neurons so that ‘by chance’ a large enough pool of neurons are obtained which encode the changes in input. Hence, there should be a range for the ratio between I_{max}^Z and I_{sat}^Z , such that we can achieve a good performance with a small number of hidden layer neurons.

To find this desired range, we first fix a value of I_{sat}^Z / I_{max}^Z and evaluate the performance of the network on both tasks with different number L of hidden layer neurons. This performance improves initially with larger L but saturates after the L increases beyond a critical value L_{min} . To quantify the dependence of performance on the ratio of I_{sat}^Z / I_{max}^Z , we now plot in Fig. 5.11 the dependence of L_{min} on the ratio of I_{sat}^Z / I_{max}^Z , with lower values of L_{min} being preferable. From this figure, the ratio of $I_{sat}^Z / I_{max}^Z \approx 0.75$ is the best trade off point between number of hidden neurons and input dynamic range for all values of σ_{V_T} . For small values of σ_{V_T} , the performance degrades rapidly on both sides of the optimal value. However, as σ_{V_T} increases, the performance degradation is much less implying the choice of I_{max}^Z is less critical in highly

Figure 5.12: Variations of classification accuracy with the resolution of output weight β showing 10 bits is sufficient for accurate classification.

scaled VLSI.

5.4.4.2 Resolution of Output Weight

As mentioned earlier, the digital circuits will use pre-calculated output weights, β from a memory and accumulate it based on neuronal spiking patterns. In order to implement this, we need to know how many bits are needed to represent β . Less number of bits will degrade performance of the classifier while more will waste hardware resources and power. We use the classification example here with $L = 128$. Figure 5.12 shows the change of error with increasing number of bits indicating 10 bits resolution is enough for good accuracy.

5.4.4.3 Counter resolution

Besides the resolution of β , we also analyzed the dependence of performance on the output counter resolution b in equation (5.11). Since we estimate the spiking frequency by using a counter to count the number of spikes in a fixed time window T_{neu} , a small value of b will introduce large quantization errors in the estimate of frequency. This implies that the neurons

Figure 5.13: Variations of classification accuracy with the number of bits of counter output H demonstrating that $b \approx 6$ is enough for optimal performance. Each of the curves are averaged over 50 trials.

have to produce more spikes in the counting window, which would on the other hand induce more power dissipation. To find a good trade-off for b , we fixed $I_{sat}^z/I_{max}^z \approx 0.75$, $L = 128$ and resolution of β to 10 bits. Figure 5.13 shows the simulation result for the classification error with b increasing from 1 to 10. $b \approx 6$ is found to be sufficient for classification.

5.5 Noise, Speed and Energy dissipation

5.5.1 Noise

Noise is an important specifications to be considered in circuit design In this section, we present the operational limits set on this architecture due to noise based constraints. Since the transistors are operating in sub-threshold region, the contribution of $1/f$ noise is negligible compared to the thermal noise [11]. For the current mirror circuit as shown in Fig. 5.10, we can

easily get the input referred thermal noise spectral density as:

$$\overline{i_{in}^2} = \overline{i_{n1}^2} + \overline{i_{n2}^2} \cdot \frac{g_{m1}^2}{g_{m2}^2}, \quad (5.13)$$

where g_{m1} and g_{m2} are transconductance of input and output transistors respectively, $\overline{i_{n1}^2}$ and $\overline{i_{n2}^2}$ are corresponding transistor channel noise. Since the transistors are working in the sub-threshold region, the transconductance is in proportion to its drain current. Applying the noise model of drain current of sub-threshold transistors to be $\overline{i^2} = 2qI\Delta f$ [89] where q denotes electronics charge, we can rewrite the above equation as:

$$\overline{i_{in}^2} = 2qI_1\Delta f + 2q\Delta f \cdot \frac{I_1^2}{I_2} \quad (5.14)$$

For this single pole system, the noise equivalent bandwidth $\Delta f = \frac{\kappa I_1}{4CU_T}$ where κ denotes the inverse of the sub-threshold slope [89]. Assuming $I_2/I_1 = w_0$, and substituting the bandwidth equation above, we get:

$$\overline{i_{in}^2} = \frac{q\kappa I_1^2}{2CU_T} \left(1 + \frac{1}{w_0} \right). \quad (5.15)$$

Finally, the signal to noise ratio (SNR) can be expressed in the following equation:

$$SNR = \frac{I_1^2}{\overline{i_{in}^2}} = \frac{2CU_T w_0}{q\kappa(w_0 + 1)}. \quad (5.16)$$

Thus, from the equation (5.16), we can see the SNR can be controlled by changing C . This reflects a direct trade-off with bandwidth which is inversely proportional to C . If an 8 bits SNR is needed in the system, and $w_0 = 1$, it is sufficient to add $C = 0.4\text{pF}$ capacitance in the current mirror for each input channel. Note that only one such capacitor is needed for every row.

5.5.2 Speed

The conversion time for one classification operation T_c comprises two parts: T_{cm} and T_{neu} where T_{neu} is the neuron operation time and T_{cm} is the current mirror settling time. If one of them is much larger than the other, we can approximate $T_c \approx \max(T_{cm}, T_{neu})$. We consider T_{cm} to

Figure 5.14: Using active current mirror for small input currents can boost the bandwidth by $5.84X$.

be 4 times of the inverse of the bandwidth, i.e. $T_{cm} = \frac{4}{BW} = \frac{4CU_T}{\kappa I_{in}}$ where $\kappa = 0.7$, $U_T = 0.025V$ at room temperature and $C = 0.4pF$ as derived earlier. If the average input current is $I_{max}/2$, the average current mirror settling time is

$$T_{cm,avg} = \frac{8CU_t}{\kappa I_{max}}. \quad (5.17)$$

As discussed earlier in Section 5.4.1, an active current mirror is utilized to boost the bandwidth for small current values. SPICE simulation result for this effect shown in Fig. 5.14 demonstrates a bandwidth increase by around $5.84X$. We can find the range of T_{cm} by considering maximum and minimum input currents:

$$\begin{aligned} T_{cm,max} &= \frac{4CU_t}{5.84\kappa I_{max}/2^{b_{in}}} \\ T_{cm,min} &= \frac{4CU_t}{\kappa I_{max}} \end{aligned} \quad (5.18)$$

Figure 5.15: Variation of neuron counting time (T_{neu}) and current mirror settling time (T_{cm}) reduce as maximum input current per dimension (I_{max}) is increased. Further, T_{neu} increases exponentially with increase in b .

where $b_{in} = 10$ is the number of bits of *Data_in* and the factor of 5.84 is due to the active current mirror. Figure 5.15 shows the decrease of T_{cm} with increasing I_{max} for the conventional and active current mirror cases.

To find the value of T_{neu} , we can see from Fig. 5.7(b) that we want $H = 2^b$ for $I^z = I_{sat}^z$. Combining this observation with equation (5.11), we can derive the following:

$$T_{neu} = \frac{2^b}{K_{neu}I_{sat}^z} = \frac{2^b}{0.75K_{neu}I_{max}^z} = \frac{2^b}{0.75K_{neu}dI_{max}}. \quad (5.19)$$

where we use $I_{sat}^z/I_{max}^z = 0.75$ (shown earlier in Section 5.4.4) and $I_{max}^z = d \times I_{max}$. Now, we can compare T_{cm} and T_{neu} to see the dominant term as a function of parameters b and d . Figure 5.15 shows a comparison between $T_{cm} = 0.5(T_{cm,max} + T_{cm,min})$ and T_{neu} for $b = 8$ and $b = 12$. Increasing I_{max} reduces the time required for both the neuron and current mirror. T_{cm} for the conventional current mirror is always the dominant factor. However, with the active current mirror on T_{neu} may be larger than T_{cm} for large values of b . These plots are done for $d = 10$; increasing d will have an effect of reducing T_{neu} since $I_{max}^z = d \times I_{max}$ increases. Hence, to show

Figure 5.16: Contours where T_{cm} is equal to T_{neu} in the space of counter dynamic range 2^b and input dimension d . For increasing d , the total current input to a neuron I^z keeps on increasing thus increasing oscillation frequency. Hence, it can support higher dynamic range 2^b in the same time T_{neu} .

the trade-offs between T_{cm} and T_{neu} as a function of b and d , we plot contours in the space of counter dynamic range 2^b and input dimension d where $T_{cm} = T_{neu}$. To do this, we equate (5.17) and (5.19) to get:

$$\begin{aligned} \frac{8CU_t}{\kappa I_{max}^z/d} &= \frac{2^b}{K_{neu} I_{sat}^z} \\ \implies 2^b &= \frac{6dCU_t K_{neu}}{\kappa} \end{aligned} \quad (5.20)$$

where $I_{sat}^z/I_{max}^z = 0.75$ is used. The straight line contours defined by equation 5.20 are plotted in Fig. 5.16 for three different K_{neu} values corresponding to VDD= 0.8, 1 and 1.2V. For parameter choices on these contour lines, $T_c = T_{cm} + T_{neu} = 2T_{cm} = 2T_{neu}$. If the relation between 2^b and d sets the operation regime above any of the contour lines, $T_{neu} > T_{cm}$ while the opposite condition is true if operation regime is below the contour lines. It can be seen that for $b \approx 8 - 10$ bits and a nominal value of VDD=1V, T_{neu} dominates T_{cm} for the maximum dimension of 128 supported by our chip.

5.5.3 Energy

The total power dissipated by the system (P_t) can be split into two parts: power from analog (P_{avdd}) and digital (P_{vdd}) supplies. The first term (P_{avdd}) is mainly dissipated by the voltage reference circuitry, biasing block and the IGCs. Ideally, this should be a function of input dimension. However, in the current design only unused active mirrors are turned off while the current DAC is always ON—this will be rectified in future designs. The second term (P_{vdd}) comprises the power dissipated by the neuron, asynchronous counter and other digital blocks including decoder and scanner. Of these terms, the power dissipated by the neuron includes the synaptic currents as the input and the counter at output and varies with different parameters such as biasing current. It is the major energy consumer in the chip when the number of hidden neurons, L is large. Hence, it is important to understand its dependence on different parameters. Thus, we can write P_{vdd} as:

$$P_{vdd} = P_{neu} + P_{cnt} + P_{dig} \approx P_{neu} = Lf_{sp}E_{sp}, \quad (5.21)$$

where E_{sp} is the energy dissipation per spike for the neuron. The power dissipated by the neuron includes the synaptic currents as the input and is the major energy consumer in the chip; hence, it is important to understand its dependence on different parameters. E_{sp} can be modelled as:

$$E_{sp} = \alpha_1 VDD^2 + \frac{\alpha_2 I_{sc} VDD}{f_{sp}} + \frac{C_b I^2 VDD^2}{I_{rst} - I^2 + I_{lk}}, \quad (5.22)$$

where I_{sc} is the short-circuit current in the inverter that depends on the value of VDD and is negligible for small values of VDD. Here, the first term denotes the switching power dissipated in the neuron circuit, second term denotes short circuit power loss in the inverters and the third term denotes the short-circuit power dissipated on the node V_{mem} in Fig. 5.5. If $I^2 \ll I_{rst}$ and $I_{lk} \approx 0$, equations (5.21) and (5.22) can be combined to give:

$$P_{vdd} \approx P_{neu} \approx L (\alpha_1 VDD^2 f_{sp} + \alpha_2 I_{sc} VDD). \quad (5.23)$$

From simulation, when VDD is 1V, $\alpha_1 \approx 0.2pF$ and $\alpha_2 I_{sc} \approx 0.03\mu A$.

Using equation (5.22), we will now proceed to estimate average energy per conversion operation (E_c) for one neuron where an input current $I^z \in [0 I_{max}^z]$ is converted to a digital count. Assuming that I^z is distributed uniformly in the range of 0 to I_{max}^z , i.e. $P(I^z) = \frac{1}{I_{max}^z}$, E_c can be estimated as:

$$\begin{aligned} E_c &= \int_0^{I_{max}^z} E_{sp}(I^z) H(I^z) P(I^z) dI^z \\ &= \frac{1}{I_{max}^z} \int_0^{I_{max}^z} E_{sp}(I^z) H(I^z) dI^z, \end{aligned} \quad (5.24)$$

where $H(I^z)$ is the number of spikes generated in T_{neu} as defined in equation (5.11). Note that here we write $E_{sp}(I^z)$ and $H(I^z)$ to make the dependence of equations (5.22) and (5.11) on I^z explicit. Using the expression for T_{neu} in equation (5.19), equation (5.24) can be simplified further to get:

$$E_c = \frac{2^b}{0.75K_{neu}I_{max}^z{}^2} \int_0^{I_{max}^z} E_{sp}(I^z) f_{sp}(I^z) dI^z. \quad (5.25)$$

From equation (5.25), we can see that E_c depends on I_{max}^z . The choice of I_{max}^z is guided by the design constraints. Typically, we have to either meet a minimum specified speed of operation or minimize energy of operation without any constraint on speed. To better explain the trade-offs, we can plot E_c while varying I_{max}^z with $b = 10$ as illustrated in Fig. 5.17 for three values of VDD. The same figure is re-plotted in Fig. 5.18 but with the corresponding value of T_{neu} instead of I^z . Firstly, note that the plots for smaller VDD span a smaller range of current since I_{rst} is correspondingly smaller (similar to Fig. 5.9). For each VDD, the lowest conversion energy is attained when I_{max}^z is close to $I_{flx} = I_{rst}/2$. Intuitively, this happens because f_{sp} is higher which leads to lower T_{neu} and correspondingly lower energy. Thus it is beneficial to operate for a short time at a higher spiking frequency than over a longer time with a small frequency. The optimum current I^z is less than I_{flx} since at $I^z = I_{flx}$, the short-circuit power dissipation (third term in equation (5.22)) increases significantly. From Fig. 5.17 and 5.18, we

Figure 5.17: Variation of energy per classification operation (E_c) with varying maximum value of input current I_{max}^Z for three different settings of VDD.

can see that lowest energy per conversion is attainable for lowest VDD as expected since the short circuit current reduces drastically at lower VDD. However from Fig. 5.18, we can see that the trade-off for keeping a low VDD is large conversion time. Hence, if conversion time is a critical specification, we have to choose the minimum VDD that meets this specification. As can be seen from Fig. 5.18, higher VDD allows for lower T_{neu} .

5.6 Input Dimension and Hidden Layer Extension Technique

For some applications, dimension of the input data is quite large (over several thousands) while other applications may require a large number of hidden layer neurons (also over several thousands) to achieve the best performance. This poses a big challenge to neuromorphic analog hardware implementations and have restricted the use of analog classifiers since the dimensions of the chip are fixed once fabricated. For example, suppose the input-dimension for an application is d and it requires L hidden layer neurons. Conventionally, at least $d \times L$ random weights are needed for the random projection operation in the first layer of ELM to get the hidden layer

Figure 5.18: The same plot as in (a) but replacing I_{max}^z with its corresponding T_{neu} from equation (5.19).

matrix $\mathbf{H}$. However if the maximum input dimension for the hardware is only k ($k < d$) and the number of implemented hidden layer neurons is N ($N < L$), the hardware can only provide a $k \times N$ random projection matrix $\mathbf{W}$ comprising weights w_{ij} ($i = 1, 2, \dots, k$ and $j = 1, 2, \dots, N$). For more efficient use of the hardware, here we propose a method to reuse the input weights and hidden layer neurons to effectively expand both input dimension and number of hidden layer neurons beyond the number physically fabricated on-chip. Intuitively, each neuron requires d random weights and there are a total of $k \times N$ such random weights on the chip. Hence, as long as $d < k \times N$, we can reuse these random weights to satisfy the requirement. Similarly, each input dimension requires L random numbers for the projection—it can be attained by reusing weights as long as $L < k \times N$. A simple example of such an increased dimension of weight matrix is shown in Fig. 5.19 for $k = 2$ and $N = 3$. This case shows the maximum dimension increase possible to get a matrix of size $(k \times N) \times (k \times N)$. Next, we elaborate the method used to do this assuming $d, L < k \times N$.

To expand the number of hidden layer neurons, we propose to do it in $\lceil L/N \rceil$ steps where the number of projections is increased N in every step. For the second set of N neurons, we need to

Figure 5.19: The extension from a 2×3 random projection matrix to 6×6 by weight reuse technique .

shift the random matrix $\mathbf{W}$ comprising w_{ij} ($i = 1, 2, \dots, d$ and $j = 1, 2, \dots, N$) to $\mathbf{W}_{1,0}$ comprising w_{ij} ($i = 2, 3, \dots, d, 1$ and $j = 1, 2, \dots, N$). Here, the subscript $(1,0)$ is used to denote a single circular rotation of the rows of the matrix $\mathbf{W}$. This notation implies $\mathbf{W} = \mathbf{W}_{0,0} = \mathbf{W}_{k,0}$. Using this notation, we can continue to get more random projections of the input (and thus expand the number of hidden neurons) by generating $\mathbf{W}_{1,0}$ to $\mathbf{W}_{\lceil L/N \rceil - 1, 0}$. Figure 5.20(a) shows a simple circuit that can be added to the input side of the chip to achieve this function. The corresponding timing diagram of control signals are shown in Fig. 5.20(b). Once the input data is loaded and the first set of hidden layer outputs are obtained (during the NEU_EN signal), the *Rotation_Control* signal is turned high to configure the input registers as a circular shift-register. This is followed by another NEU_EN signal to obtain the second set of N random projections and this process continues till L random projections are obtained.

A similar method can be applied to expand the input dimension from k to d . In this case, we take the first k dimensions $x_1, x_2, \dots, x_k$ of a particular input sample $x \in \mathfrak{R}^d$ and send it to the chip to get the multiplication for the first k dimensions with the random matrix $\mathbf{W}$. This generates L hidden neuron outputs which can be expanded to a larger number using the technique described in the last paragraph. For the next k dimensions of x , we shift the random matrix $\mathbf{W}$ comprising w_{ij} ($i = 1, 2, \dots, k$ and $j = 1, 2, \dots, N$) to $\mathbf{W}_{0,1}$ comprising w_{ij} ($i = 1, 2, \dots, k$ and $j = 2, 3, \dots, N, 1$). This implies a circular shift along the columns of $\mathbf{W}$. The hidden layer outputs obtained in this step are added to the ones obtained in the earlier step. This method can be continued for

Figure 5.20: (a) Schematic of peripheral circuit for hidden layer extension by shifting the input data stored in the registers and (b) its timing diagram.

$\lceil d/k \rceil - 1$ steps while accumulating the resulting hidden layer outs every time to get the final output for the d dimensional input x . Figure 5.21(a) shows a simple circuit that can be added to the previously described chip architecture at the output to implement the input dimension expansion technique. Figure 5.21(b) depicts the corresponding timing diagram. The circuit in Fig. 5.21(a) shows a register bank after the neuron output counters that can accept inputs from these counters or from other registers in this layer to effect the circular rotation of columns of $\mathbf{W}$. There is a second register bank after this which accumulates the counter outputs over multiple cycles. After the conversion of first k dimensions of x during the first *NEU_EN* signal, a clock pulse on *CLK_r* and *CLK_a* are used to shift this output to the accumulator. From the next cycle, the *Rotation_Control* signal is enabled and pulses on *CLK_r* are used to rotate the

Figure 5.21: (a) Schematic of circuit for input dimension extension by shifting and summing the output counter values and (b) its timing diagram.

columns of the hidden layer. Another pulse on CLK_a is used to accumulate this value in the second register bank.

5.7 Measurement Results

5.7.1 Characterization

To validate the function of the proposed design, we have implemented the system in a $0.35\mu\text{m}$ CMOS process. The ELM chip occupies a die area of $5\text{mm} \times 5\text{mm}$ as shown in Fig. 5.22. The current area of the chip is dominated by the current mirror array since the layout is not

Figure 5.22: Die photo of the prototype chip fabricated in $0.35\mu\text{m}$ CMOS.

optimized. Each cell in the current mirror array is pitch matched to the neuron in one direction and the IGC along another making it mostly empty. The area of the current mirror array can be reduced tremendously by following the proposal in [64] limiting the size to the pitch of the IGC. In the next version, we will reduce the pitch of the IGC by moving to a scaled process like 65nm . The mixed-signal chip implements the computationally intensive first stage while the second stage is currently implemented off-chip on a FPGA. In future, the second stage will also be integrated on the same die. Again, moving to a scaled process like 65nm enables a small layout for this digital part. The larger statistical variation in a scaled process does not hurt the performance of the analog part as shown in Fig. 5.11, Fig. 5.12 and Fig. 5.13. The extra gate leakage in the current mirrors can be handled by either using thick oxide I/O devices or using active mirrors. Next, we present some characterization results to show the functionality of the chip. In all the experiments, both analog and digital power supplies are shorted together and

Table 5.2: Chip Summary

Technology	$0.35\ \mu\text{m}$ CMOS
Die Size	$5\ \text{mm} \times 5\ \text{mm}$
Input Channels	128
Hidden Layer Size	128
Output Data format	14-bit Digital
Input Data format	10-bit Digital
Power supply voltage	1 V

Figure 5.23: Measured transfer function of hidden layer neurons when the digital input varying from 0 to 1023 with $d = 1$ and $T_{neu} = 10\text{ms}$.

Table 5.3: Mean and standard deviation of ΔV_{Tn}

Chip #	μ (mV)	σ (mV)
1	-0.20	15.91
2	-0.16	15.36
3	-0.35	15.68
4	-0.28	15.70
5	-0.68	16.03
6	-0.15	15.77
7	-0.53	16.19
8	-0.15	15.61
9	-0.48	16.26

is denoted by VDD. Unless stated otherwise, the default value of VDD= 1V is used in most experiments.

First, we can get the transfer function of the 128 neurons by sweeping the digital input $Data_in$ on any one channel from 0 to 1023. The resultant curves are shown in Fig. 5.23. It can be seen that there is significant variation between the transfer curves of the neurons. Next, to characterize the random variation of the input weight matrix, we send a fixed value of $Data_in$ to each of the input channels one by one and measure the counter outputs H . For every input channel, we get $L = 128$ counter values indicative of the mismatch in that row.

Figure 5.24: A surface plot showing the mismatch in weights of the 128×128 current mirror synapses. The output counter values for different neurons are plotted for $T_{neu} = 10\text{ms}$ when $Data_in = 100$ is set on each input channel one by one.

In total, there are 128×128 such values of H for all the input channels. These results are shown as a 3-dimensional plot in Fig. 5.24 where H is plotted on the Z-axis. These same values are normalized by the median count value to get the effective weight distribution. This distribution of 128×128 values is plotted as a histogram in Fig. 5.25 displaying a log-normal distribution. This is to be expected since ΔV_{Tn} has a normal distribution as explained in Section 5.4.3. Further, by fitting a gaussian distribution to the logarithm of the weight values, we obtain $\sigma \Delta V_{Tn} \approx 16\text{mV}$ in this process. Note that the mismatch obtained here also takes into account mismatch in the neuronal tuning curves since the count values are obtained at the output of the neuron. Further, this characterization is consistent across a set of 9 chips as shown in Table 5.3.

5.7.2 Speed and Power

During measurement, we found the chip to be functional for VDD down to 0.7 V. Thus we can apply the results of the design space exploration in Section 5.5 to optimize the system for the best speed and power efficiency. During measurement, a pico-ammeter (Keithley 6485) is utilized to measure the average current from the power supply to estimate the power dissipation.

Figure 5.25: Histogram showing the log-normal distribution of the input weights obtained from 5.24 for the 128×128 current mirror array.

For all the experiments, speed and power are measured for $Data_in = 1000$ and $d = 128$ with $L = 100$ neurons activated. Conversion times T_{neu} are estimated for $2^b = 128$. At VDD= 0.7V, the power dissipation is $17.85\mu W$ at a maximum conversion speed of 4.5kHz. As can be expected from Fig. 5.17, there is not much variation in energy per classification when I_{max}^z is reduced. However, this difference is more obvious at a higher VDD of 1V. In this case, the fastest classification rate for this system is 146.25 kHz corresponding to $T_{neu} = 68.5\mu s$ when $I^z \approx I_{flx}$. However, the power dissipation at this speed is quite high—2.2mW. Hence for a better energy efficiency, we optimize the classification rate to be around 31.6 kHz by reducing I_{max}^z to reduce the short-circuit power dissipation on V_{mem} (as described in Section 5.5.3). The measured power dissipation now becomes $188.8\mu W$ as shown in Table 5.5. We choose this operating point as a good trade-off between speed and power efficiency. From this, we can approximate the coefficients $\alpha_1 \approx 0.3pF$ and $\alpha_2 I_{sc} \approx 0.076\mu A$ that are close to simulation values reported in section 5.5.3. Also, the analog power $P_{avdd} \approx 3.4\mu W$. Considering the 128×100 multiplication-and-accumulation (MAC) operation for the first layer, we can calculate the energy efficiency for this case as 0.47 pJ/MAC. The corresponding throughput for classification

Figure 5.26: Regression of underlying sinc function (in blue) based on a set of noisy samples (in green).

rate of 31.6 kHz is 404.5 MMAC/s.

5.7.3 Regression and Classification

In order to verify the performance of the proposed neuromorphic ELM system in machine learning applications, we first show an example of regression ($d = 1$) where the system was trained on 5000 noisy samples (additive gaussian noise with $\sigma = 0.2$) of a target $\text{sinc}(x)$ function and its task was to approximate the underlying function through regression. The measured result of this experiment are shown in Fig. 5.26 for $L = 128$ hidden neurons where the noisy samples are shown in green and the regressed function is in blue. The error of 0.021 we obtain in this experiment is comparable to the error of 0.01 obtained in software simulations of ELM [86].

Next, we employ some real-world benchmark binary classification data sets from the UCI machine learning repository [84]. The reason for choosing these data sets are that they have different characteristics in terms of data dimension d and data set size in terms of number of samples: small size and low dimensions (*Pima Indians diabetes*, *Statlog Australian credit*), large size and low dimensions (*Star/Galaxy – Bright*), large size and high dimensions (*Adult*).

Table 5.4: Measured performance on Binary Classification Datasets from UCI repository

Datasets	# Features (d)	# Training	# Testing	Miss Classification Rate (%)	
				Software [48] ($L = 1000$)	This work ($L = 128$)
Diabetes	8	512	256	22.05	22.91
Australian Credit	14	460	230	13.82	12.11
Brightdata	14	1000	1462	0.69	1.26
Adult	123	4781	27780	15.41	15.57

The details of the data sets are shown in Table 5.4. During measurements, the hidden layer matrix $\mathbf{H}$ is obtained by applying the training data to the chip one by one. The second layer weights are obtained offline using this $\mathbf{H}$ and then downloaded to the FPGA for testing. The accuracy obtained in measurements with $L = 128$ hidden neurons is shown in table 5.4 and is compared with software simulation results taken from [48]. This table shows that the performance of our implemented hardware ELM is comparable with the software ELM with the differences possibly due to the larger number of sigmoidal neurons (as opposed to saturating linear neurons for this chip) used in [48].

5.7.4 Dimension Increase With Weight Reuse Technique

In order to evaluate the performance for the dimension extension technique, we first applied a very high dimensional dataset (*leukemia*) with $d = 7129$. Sizes for the training and testing data are 38 and 34 respectively. During measurement, we obtain a miss-classification rate of 20.59% with $L = 128$ neurons, which is comparable with the error rate of 19.92% obtained using the software ELM reported in [48]. Next, we separately prove the concept of artificially increasing number of hidden layer neurons. The measured errors in table 5.4 are close to optimal and do not reduce much with further increase in L . Hence, we instead take $L = 16$ neurons and use weight reuse method to expand to $L = 128$. For the dataset *diabetes*, the error for $L = 16$ is 27.1%. This reduces to an error of 22.4%, comparable to that in table 5.4, when L is increased to 128 by weight reuse. Note that since our chip did not have the circuits described in Section 5.6 to perform on-chip dimension expansion, we shifted the input data before applying it to the chip. Also, the output data was shifted in the FPGA before accumulation.

5.7.5 Comparison

Our work is compared with other recently reported hardware machine learners in Table 5.5. Our design is the most power efficient machine learner reported so far due to the low power analog multiplications. The energy efficiency of commercial digital processors are saturating at $\approx 100\text{pJ/MAC}$ [90]. Even custom digital multipliers have energy efficiencies of $10 - 70\text{pJ/MAC}$ [83, 91, 92]. This explains the higher energy requirement of [10] in Table 5.5. [11] uses analog floating-gate based multipliers and can hence achieve low-power multiplication. However, our approach does not require high voltages for programming floating-gates and is also much more compact due to the use of only one transistor without capacitors in the multiplier cell. [93] also uses random mismatch (and a systematic offset) in 65nm CMOS to perform the calculations in the first stage of ELM. However, they only have a single dimensional input and only show regression. Moreover, they do not report any energy or speed metrics. Lastly, compared to [64] which also uses the same core circuit of current mirrors to perform ELM computations for neural decoding, the current work is more energy efficient due to the faster operation (as explained in section 5.5.3). Also, the current work shows a method of expanding input dimension to a maximum of $d = 16,384$ while [64] could only support a maximum of $d = 128$. In Table 5.5, the reported power efficiency of 0.47pJ/MAC is only for the first stage of ELM. Assuming a digital processor with 10 pJ/MAC [91, 92] used to implement the second stage of ELM for binary classification, the energy per operation increases to 0.54 pJ/MAC .

5.7.6 Robustness

It is important to consider how the performance of the chip varies in the face of variations of power supply voltage (VDD) and temperature. We use the normalization method suggested in [64] to increase the robustness of our chip with respect to common-mode variations in VDD

Table 5.5: Comparison Table

	JSSC 2013 [10]	JSSC 2007 [11]	IJCNN 2015 [93]	ISCAS 2015 [64]	This work
Technology	0.13 μm	0.5 μm	65 nm	0.35 μm	0.35 μm
Algorithm	SVM	SVM	ELM	ELM	ELM
Task	Classification	Classification	Regression	Regression Classification	Regression Classification
Design Style	Digital	Analog Floating gate	Mixed mode	Mixed mode	Mixed mode
Supply Voltage	0.85 V	4 V	1.2 V	0.6 V (Digital) 1.2 V (Analog)	1 V
Power Dissipation	136.5 μW	0.84 μW	-	0.4 μW	188.8 μW ¹
Max Input Dimension	400	14	1	128	16384 ²
Energy Efficiency	631 ³ pJ/MAC	0.8 pJ/MAC	-	3.4 ⁴ pJ/MAC	0.47/0.54 ⁵ pJ/MAC
Resolution	16 b	4.5 b	13 b	14 b	14 b
Classification Rate	0.5-2 Hz	40 Hz	-	50 Hz	31.6 kHz
Throughput	2 MMAC/s	1300 MMAC/s	-	0.12 MMAC/s	404.5 MMAC/s

¹ This power dissipation is measured based on $d = 128$ and $L = 100$.

² Using input dimension extension technique to expand to $d = 128 \times 128$.

³ Assuming 1000 support vectors.

⁴ Only considering first stage of ELM for $d = 40$ and $L = 60$.

⁵ 0.47 pJ/MAC is energy efficiency of current chip implementing first stage of ELM. The total energy per operation for binary classification is 0.54 pJ/MAC (see section 5.7.2 for details).

Figure 5.27: Comparison of hidden layer outputs for three different values of VDD in (a) the conventional case and (b) normalized case. The normalization results in less variation of output due to change in VDD.

and temperature. Following, [64], we define the j -th normalized hidden layer value ($h_{j,norm}$) as:

$$h_{j,norm} = \frac{h_j}{\sum_{j=1}^L h_j / \sum_{i=1}^d x_i} \quad (5.26)$$

To show the effectiveness of normalization, we first consider its effect on variations in VDD. Figure 5.27(a) plots measured values of hidden layer output h_j for five different values of input data D_{in} at three different values of VDD (0.8, 1 and 1.2V). It can be seen that there is a huge variation in h_j (maximum of 22.7%). In contrast, when the same values are normalized

(a)

(b)

Figure 5.28: Comparison of performance when normalized and non-normalized hidden layer outputs are used for classification of (a) Australian credit and (b) Brightdata sets from the UCI repository.

(Fig. 5.27(b)), the variation due to change in VDD is reduced a lot (maximum of 4.2%) while variation due to change of D_{in} is still retained. This proves effectiveness of the normalization method. We have further used the normalized and non-normalized values to perform the *sinc* function regression task described in Section 5.7.3. In this case, the weights are obtained for a nominal VDD of 1V while testing is performed at all three VDD values. The result is reported in Table 5.6. It can be seen that normalization enables the error to be low for all three values of VDD.

Next, we studied the effect of temperature variations on the hidden layer outputs. We expect

Table 5.6: *Sinc* function regression using normalized h_j

Power supply (V)	Error (%) (Non-normalized)	Error (%) (Normalized)
0.8	0.5924	0.076
1	0.045	0.0629
1.2	0.1538	0.065

the temperature dependent weights ($e^{\frac{\Delta V_T}{U_T}}$) to be the major contributor to variations in hidden layer outputs h_j . To confirm this prediction, we made a MATLAB model and obtained the variation of h_j when temperature varied by $\Delta T = \pm 20^\circ\text{C}$ about a nominal value of $T_0 = 300\text{K}$. Then we benchmarked this variation with a SPICE simulation of the same circuit to confirm our earlier assumption—henceforth, we used the MATLAB model for simulations. Similar to the earlier case, we found that applying normalization reduced the maximum variation of hidden layer outputs from 9% to 1.6% over this temperature range. Next, we trained output weights for classification problems at the nominal temperature T_0 while the temperature was again varied over the same range during testing. We plot the results for h_j and $h_{j,norm}$ for two different datasets in Fig. 5.28(a) and (b). It can be seen that the error increases rapidly when temperature varies on either side of T_0 while using h_j . On the other hand, the error changes much more slowly when using $h_{j,norm}$ again confirming the benefit of normalization. Further, we have observed that retraining the weights can reduce the error close to the original value for both h_j and $h_{j,norm}$. Hence, to get good performance over a wider range of temperature, we can store different weights for different temperature ranges. One disadvantage with using the normalization is that now the second layer has to perform L divisions on top of the $L \times C$ multiplications. But given the benefits provided, we believe that normalization is still a favourable choice.

5.8 Summary

We have presented a low-power hardware neuromorphic IC in $0.35\mu\text{m}$ CMOS for machine learning applications. The particular algorithm we employed is extreme learning machine (ELM). The mismatch in silicon spiking neurons and synapses are used to perform the vector-

matrix multiplication that forms the first stage of this classifier and is the most computationally intensive. A design space exploration is performed to show that minimum energy per operation at a specific VDD is obtained by operating for a short time at the highest spiking frequency achievable at that VDD. Linear neurons with a saturating non-linearity are used due to ease of implementation. Operating from a 1 V power supply, this system can achieve an optimum energy efficiency of 0.47 pJ/MAC with a corresponding classification rate of 31.6 kHz making it one of the most energy efficient machine learners reported. We also show a normalization method that enables a more robust operation of the circuit over changes in power supply and temperature.

The features we get from the feature extraction block can also be directly fed to the ELM classifier since it has a 10-bits SPI interface at the input. Each feature could take one of the input channel of the classifier. No extra circuit is needed. Hence, there would not be more resource induced. The performance of spike sorting with our ELM classifier is shown in [94]. From measurement, we can see that using ELM based classification has a better performance compared to the results using superparamagnetic clustering and K-means.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Biomedical smart sensors can be used in a wide range of applications to help improve the health condition of our human being. This thesis focuses on the design of low-power smart sensor circuits for biomedical applications, like neural recordings or any other biomedical devices.

In the area of neural recordings, we proposed a novel low power, compact, current-mode spike detector circuit for real-time neural recording systems where neural spikes or action potentials are of interest. To compress data for applications requiring only spike timing information, we can just transmit the neural spike timing information instead of transmitting the raw data digitized by the ADC. It will reduce the data rate to be transmitted by 10 to 12 times. Since spikes are sparsely distributed in the neural signal, the output of the SPD might be zero for most of the time. The long stream of zeros can be effectively compressed by schemes like run-length encoding resulting in huge compression ratios of 50 – 5000 depending on spike frequency at input.

The SPD saves power by approximating NEO using simple circuits and also uses current mode design enabling lowering of supply voltage. We also show that an integrated low pass filter with cut-off frequency $\approx 50 - 70$ Hz can be used to either track and reject low frequency interference from LFP (mode I) or provide a firing rate independent estimate of background noise for automatic threshold setting in SPD (mode II). We presented detailed simulations and

measurements showing that the performance of our design is comparable with recent publications and software simulations of traditional NEO. The chip has been fabricated in a 65 nm CMOS process and the SPD occupies a die area of $200\ \mu\text{m} \times 150\ \mu\text{m}$. Operating from a 0.7V supply, the SPD dissipates only 30 nW of static power per channel and 0.07 nJ of dynamic energy per input spike making it the lowest power consuming SPD reported till date. To reduce sensitivity of threshold setting, this work uses a current-mode oscillator based detection probability estimator (DPE) to reject false positives caused by the background noise.

Another sub-block in the neural recordings is spike sorting, whose function is to cluster the recorded neural spikes and assign them to the individual neurons. It is an essential part of the signal processing chain for neural recording system and lays the foundation for neural signal decoding. For spike sorting, the action potentials will be detected first, which is already discussed above. Once a spike is detected, features will be extracted for clustering. The classification engine for spike sorting can implement any machine learning algorithm, for example extreme learning machine (ELM) or support vector machine (SVM). After the sorting, the generated spike trains are passed to a second machine learner for decoding intention. For example, in a motor prosthetic, we might decode the intention of the user to move his or her fingers and wrist.

The proposed spike based feature extraction circuit can share the same differentiator and current rectifier. In addition, it utilizes the current-controlled oscillator to generate digital pulse trains as features of the neural spike signal for clustering. The advantage of spike based feature extraction is that the feature spikes can be directly used for neural signal decoding—there is no need for an explicit step of spike sorting. Hence, only one machine learner is enough in the neural recording for both spike sorting and decoding. In this work, we do not show the decoding—rather we just show that there is sufficient information in the spike based features to even sort the original neural signals explicitly. The spatio-temporal pattern of the pulses reflects the shape of the neural spike waveform. This method has already been utilized to distinguish the spike signal from the background noise in the design of detection probability estimator in SPD. The spike for each waveform are counted in short time windows and a feature vector is made by concatenating several such counts. K-means clustering algorithm is utilized as the classification

engine in Matlab. Simulated neural signal with various noise level have been applied to evaluate the performance and it is shown that the performance of spike based features is slightly worse than PCA based features extracted from the original waveform.

In this thesis, we also proposed a compact low-power, high performance hardware implementation of ELM which can be utilized as a co-processor in the smart sensors for biomedical applications. The proposed circuit utilizes device mismatch prevalent in today's VLSI process to perform a significant part of the computation while a digital back end enables precision in the final output. Mismatch in silicon spiking neurons and synapses are used to perform the vector matrix multiplication that forms the first stage of this classifier and is the most computationally intensive. A mixed-signal machine learning IC is implemented in a $0.35\ \mu\text{m}$ process which can be configured for a wide range of applications with 10 bits digital input. The die size of this chip is $5\ \text{mm} \times 5\ \text{mm}$. Maximum input dimension is 128 and the number of hidden layer neuron is also 128. The output of the ELM IC can be interfaced with an FPGA for second stage signal processing. A wide range types of dataset are applied to system for evaluation of the function. A design space exploration involving speed, power and accuracy is performed. Some techniques to improve performance in VLSI random projection networks is also proposed. For more efficient use of the hardware, we proposed a method to reuse the input weights and hidden layer neurons. For some applications, requirement for input dimension or hidden layer exceeds the size of implemented hardware. In this case, we can rotate and extend the random projection matrix from $d \times L$ to $dL \times dL$. Operating from a 1 V power supply, this system can achieve a classification rate of 31.6 kHz with the energy efficiency of 0.47 pJ/MAC.

6.2 Future Work

The research conducted over the past a few years focuses mainly on the design of some basic circuit blocks in the smart sensors for biomedical applications. We took the neural recording system as one of the examples. However, there still is a lot of work remaining to be done in the future to improve the performance and efficiency for the smart sensors for biomedical applications.

With the development of micro-electrode-array (MEA), signal simultaneously recorded from hundreds of channels has been possible for the neuroscientist, which is an inevitable process of brain machine interfaces. Recent works are trying to transmit the recorded neural signal wirelessly to the workstation or a PC for processing to get useful information. The big problem is how to transmit and process the acquired tremendous data especially in the future when we want to scale such systems to record from millions of neurons. In the future, with the development of smart sensor circuits, instead of transmitting the recorded raw data, the large amount of data could be compressed locally before transmission. Neuromorphic machine learner is just one such kind of circuit which can be integrated together and directly interfaced with the neural recording system. Local signal processing is carried out and relevant information is extracted which might be directly utilized to control the external modules, for example a prosthetic limb or a wheelchair or a computer cursor. This would be a very big system with stringent requirement on the power dissipation, which poses a challenge for the designers to reduce the power consumption for each sub-system to allow long term usage with infrequent recharging of batteries. Another option is to develop a very high efficiency energy harvesting system to provide sufficient power needed for the whole system to run almost perpetually without battery recharging.

For neural recording system, we have proposed a novel spike based feature extraction method for both spike sorting and intention decoding. As a next step, an intention decoder can be developed using the extracted feature spikes. To evaluate the function and performance of our proposed method, a custom IC for neural signal decoding needs to be designed and measurements will be conducted in the lab. Then we can try to implement a complete neural recording system with analog front-end, spike detection and sorting, neural signal decoding integrated together in a single SoC. An on-chip ADC block is also needed for the user to observe the raw data for system set-up or debug. However, this ADC can be turned off for most of the time once the system is already fully set-up. The machine learner used in the neural recording system can be a ELM. Since we have successfully implemented the ELM in the hardware for a variety of applications, we can utilize this ELM in the process of neural recording for spike

sorting and intention decoding. Measurement have been conducted for spike sorting using our proposed ELM chip [94]. In the future, an integration of spike detection, spike sorting and machine learner based classifier can be designed for the neural recording system. Power efficiency is one of the most important issues to consider for such a complicated system. Hence, we have to minimize the power dissipation for every sub-block in the design. In future, in-vitro and in-vivo studies have to be performed on this system. Also, the issue of signal degradation in chronic implants can be tackled by incorporating online learning on the machine learner.

For the machine learner IC, we have chosen the inputs as digital signals particularly to alleviate interfacing issues with earlier blocks since analog signal interfacing suffers from loading issues that have to be considered carefully. Hence, the output of an ADC can be easily transferred to the input of the machine learner. In a fully integrated system, it is more efficient to perform analog feature extraction and feed it directly to the analog core of the machine learner bypassing the digital input and DAC. This would improve energy efficiency further. In that case, if the analog features (such as peak, trough etc) are extracted in voltage domain, we would need transconductor blocks to convert the voltages to currents before feeding it to the current mirror arrays in the machine learner. Also, another possibility is to not have the first machine learner for spike sorting but rather use it directly for intention decoding. In that case, the input to the machine learner is still digital-but it is in the form of spikes or pulses from the spike detector. We have shown interfacing circuits for such cases-digital counters and averaging circuits-in the publication below that describes a separate chip designed for this purpose. [95]

Improving the machine learning hardware will also be a topic of future work. Researchers and engineers are also trying to integrated different type of machine learners in the biomedical smart sensors for early diagnosis or prediction. In the future, we need to keep on improving the performance and efficiency of the hardware machine learner, by exploring different algorithms and architectures. To improve the performance of hardware ELM, we can try to employ different kinds of neuron circuit for a higher speed and lower power dissipation. Other ADC architectures like ramp ADC or TDC based ADC can be considered as alternatives to the current neuronal oscillator-counter combination. In addition, we can also try to explore more application areas

of hardware machine learner system.

The big advantages for our proposed ELM classifier is the energy efficiency for high dimensional applications. Besides the neural recording, we can also apply our proposed smart sensor system to classify the EEG signals from many channels to detect epileptic seizure. Epilepsy is a neurological disorder with recurrent evoking of seizure. Typically, there would be an electrical onset several second before the clinical onset, causing the patient loss of coherence or motor control. Hence, if this electrical onset can be detected, some medication can be given to the patient to suppress the seizure. For reliability consideration, multichannel detection is quite necessary. Our proposed smart sensor with ELM classifier is a very good candidate for the seizure detection and some other works are also trying to do the seizure detection in the same way [38] [96].

APPENDIX A

Author's Publication

A.1 Journal Papers

1. **Enyi Yao**, Yi Chen, and Arindam Basu, "A 0.7 V, 40 nW Compact, Current-Mode Neural Spike Detector in 65 nm CMOS," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 2, pp. 309-318, April 2016.
2. **Enyi Yao** and Arindam Basu, "VLSI Extreme Learning Machine: A Design Space Exploration," *IEEE Transactions on Very Large Scale Integration Systems*, accepted.
3. Yi Chen, **Enyi Yao**, Arindam Basu, "A 128-Channel Extreme Learning Machine-Based Neural Decoder for Brain Machine Interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 3, pp. 679-692, June 2016.
4. A. Patil, S. Shen, **Enyi Yao** and A. Basu, "Hardware Architecture for Large Parallel Array of Random Feature Extractors applied to Image Recognition," submitted to *Neurocomputing*.

A.2 Conference Papers

1. **Enyi Yao**, Arindam Basu, "A 1 V, Compact, Current-Mode Neural Spike Detector with Detection Probability Estimator in 65 nm CMOS," *IEEE International Symposium on*

Circuits and Systems (ISCAS), pp.754 - 757, 24-27 May 2015.

2. Aritra Bhaduri, **Enyi Yao** and Arindam Basu, “Pulse-based Feature Extraction for Hardware-efficient Neural Recording Systems,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016 accepted.
3. Yi Chen, **Enyi Yao**, Arindam Basu, “A 128 channel 290 GMACs/W Machine Learning based Co-processor for Intention Decoding in Brain Machine Interfaces,” submitted to *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp.3004 - 3007, 24-27 May 2015.
4. A. Patil, S. Shen, **Enyi Yao** and A. Basu, Random Projection for Spike Sorting: Decoding Neural Signals the Neural Network Way,” *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1 - 4, Oct 2015.
5. **Enyi Yao**, Shaista Hussain, Arindam Basu and Guang-Bin Huang, “Computation using Mismatch: Neuromorphic Extreme Learning Machines,” *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 294-297, Oct. 31 - Nov. 2, 2013.
6. Dinup Sukumaran, **Yao Enyi**, Sun Shuo, Arindam Basu, Dongning Zhao and Justin Dauwels, “A Low-power, Reconfigurable Smart Sensor System for EEG Acquisition and Classification,” *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 9-12, 2-5 Dec. 2012.

BIBLIOGRAPHY

- [1] K. Najafi, K. Wise, and T. Mochizuki, “A high-yield ic-compatible multichannel recording array,” *IEEE Transactions on Electron Devices*, vol. 32, pp. 1206–1211, Jul 1985.
- [2] K. Wise, A. Sodagar, Y. Yao, M. Gulari, G. Perlin, and K. Najafi, “Microelectrodes, microelectronics, and implantable neural microsystems,” *Proceedings of the IEEE*, vol. 96, pp. 1184–1202, July 2008.
- [3] P. Campbell, K. Jones, R. Huber, K. Horch, and R. Normann, “A silicon-based, three-dimensional neural interface: manufacturing processes for an intracortical electrode array,” *IEEE Transactions on Biomedical Engineering*, vol. 38, pp. 758–768, Aug 1991.
- [4] R. Harrison and C. Charles, “A low-power low-noise cmos amplifier for neural recording applications,” *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 958–965, June 2003.
- [5] R. Harrison, “The design of integrated circuits to observe brain activity,” *Proceedings of the IEEE*, vol. 96, pp. 1203–16, July 2008.
- [6] J. Holleman, A. Mishra, C. Diorio, and B. Otis, “A micro-power neural spike detector and feature extractor in .13 μm cmos,” in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 333–336, Sept 2008.
- [7] L. Hoang, Z. Yang, and W. Liu, “Vlsi architecture of neo spike detection with noise shaping filter and feature extraction using informative samples,” in *Proceedings of the International IEEE EMBS Conference on Neural Engineering*, pp. 978–981, Sept 2009.
- [8] S. W. M. Blatt and E. Domany, “Super-paramagnetic clustering of data,” *Phys. Rev. Lett.*, vol. 76, pp. 3251–3254, 1996.
- [9] V. Karkare, S. Gibson, and D. Marković, “A 130-w, 64-channel neural spike-sorting dsp chip,” *IEEE Journal of Solid-State Circuits*, vol. 46, no. 5, pp. 1214–1222, 2011.
- [10] K. H. Lee and N. Verma, “A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals,” *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 1625–1637, July 2013.
- [11] S. Chakrabartty and G. Cauwenberghs, “A Sub-microwatt Analog VLSI Trainable Pattern Classifier,” *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 1169–1179, May 2007.
- [12] J. Lu, S. Young, I. Arel, and J. Holleman, “A 1 tops/w analog deep machine-learning engine with floating-gate storage in 0.13 μm cmos,” *IEEE Journal of Solid-State Circuits*, vol. 50, pp. 270–281, Jan 2015.

- [13] J. W. Sander, "The incidence and prevalence of epilepsy." <http://www.e-epilepsy.org.uk>, 2005. [Online].
- [14] L. F. M. E. M. Pugliatti, E. Beghi and P. Sobocki, "Estimating the cost of epilepsy in europe: A review with economic modeling," *Epilepsia*, vol. 48, no. 12, p. 2224-2233, 2007.
- [15] A. H. Shoeb, *Application of machine learning to epileptic seizure onset detection and treatment*. PhD thesis, Massachusetts Institute of Technology, 2009.
- [16] M. A. Nicolelis, "Actions from thoughts," *Nature*, vol. 409, no. 6818, pp. 403–407, 2001.
- [17] J. C. Kao, S. D. Stavisky, D. Sussillo, P. Nuyujukian, and K. V. Shenoy, "Information systems opportunities in brain-machine interface decoders," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 666–682, 2014.
- [18] B. I. Rapoport, L. Turicchia, W. Wattanapanitch, T. J. Davidson, and R. Sarpeshkar, "Efficient universal computing architectures for decoding neural activity," *PloS one*, vol. 7, no. 9, p. e42492, 2012.
- [19] Y. Chen, A. Basu, L. Liu, X. Zou, R. Rajkumar, G. S. Dawe, and M. Je, "A digitally assisted, signal folding neural recording amplifier," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 4, pp. 528–542, 2014.
- [20] R. Harrison, P. Watkins, R. Kier, R. Lovejoy, D. Black, B. Greger, and F. Solzbacher, "A low-power integrated circuit for a wireless 100-electrode neural recording system," *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 123–133, Jan 2007.
- [21] P. Watkins, G. Santhanam, K. Shenoy, and R. Harrison, "Validation of adaptive threshold spike detector for neural recording," in *Proceedings of the International IEEE EMBS Conference on Neural Engineering*, vol. 2, pp. 4079–4082, Sept 2004.
- [22] D. L. DONOHO and J. M. JOHNSTONE, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.
- [23] A. Rodriguez-Perez, J. Ruiz-Amaya, M. Delgado-Restituto, and A. Rodriguez-Vazquez, "A low-power programmable neural spike detection channel with embedded calibration and data compression," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, pp. 87–100, April 2012.
- [24] E. Koutsos, S. Paraskevopoulou, and T. Constandinou, "A 1.5 uW NEO-based spike detector with adaptive-threshold for calibration-free multichannel neural interfaces," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 1922–1925, May 2013.
- [25] Y.-G. Li, Q. Ma, M. Haider, and Y. Massoud, "Ultra-low-power high sensitivity spike detectors based on modified nonlinear energy operator," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 137–140, May 2013.

- [26] T. Wu and Z. Yang, "A multichannel integrated circuit for neural spike detection based on ec-pc threshold estimation," in *Proceedings of the International IEEE EMBS Conference on Neural Engineering*, pp. 779–782, July 2013.
- [27] M. Abeles and J. Goldstein, M.H., "Multispike train analysis," *Proceedings of the IEEE*, vol. 65, pp. 762–773, May 1977.
- [28] M. Lewicki, "A review of methods for spike sorting: The detection and classification of neural action potentials," *Network: Comput. Neural Syst.*, vol. 9, p. R53R78, 1998.
- [29] Z. Yang, Q. Zhao, and W. Liu, "Spike feature extraction using informative samples," in *NIPS*, pp. 1865–1872, Citeseer, 2008.
- [30] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural computation*, vol. 16, no. 8, pp. 1661–1687, 2004.
- [31] U. Wolf, "Comparison between cluster monte carlo algorithms in the ising spin model," *Phys. Lett. B*, vol. 228, pp. 379–382, 1989.
- [32] S. W. M. Blatt and E. Domany, "Data clustering using a model granular magnet," *Neural Computation*, vol. 9, pp. 1805–1842, 1997.
- [33] R. Chandra and L. M. Optican, "Detection, classification, and superposition resolution of action potentials in multiunit single-channel recordings by an on-line real-time neural network," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 5, pp. 403–412, 1997.
- [34] S. Gibson, J. W. Judy, and D. Marković, "Comparison of spike-sorting algorithms for future hardware implementation," in *Proceedings of the International IEEE EMBS Conference on Neural Engineering*, pp. 5015–5020, IEEE, 2008.
- [35] D. Anguita, A. Boni, and S. Ridella, "A digital architecture for support vector machines: theory, algorithm, and fpga implementation," *IEEE Transactions on Neural Networks*, vol. 14, pp. 993–1009, Sept 2003.
- [36] K. Kang and T. Shibata, "An on-chip-trainable gaussian-kernel analog support vector machine," *IEEE Transactions on Circuits and Systems I*, vol. 57, pp. 1513–1524, July 2010.
- [37] R. Genov and G. Cauwenberghs, "Kerneltron: support vector "machine" in silicon," *IEEE Transactions on Neural Networks*, vol. 14, pp. 1426–1434, Sept 2003.
- [38] J. Yoo, L. Yan, D. El-Damak, M. Altaf, A. Shoeb, and A. Chandrakasan, "An 8-Channel Scalable EEG Acquisition SoC With Patient-Specific Seizure Classification and Recording Procseor," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 214–28, Jan. 2013.
- [39] T.-W. Kuan, J.-F. Wang, J.-C. Wang, P.-C. Lin, and G.-H. Gu, "Vlsi design of an svm learning core on sequential minimal optimization algorithm," *IEEE Transactions on VLSI*, vol. 20, pp. 673–683, April 2012.

- [40] T. Serrano-Gotarredona, B. Linares-Barranco, and A. Andreou, "A general translinear principle for subthreshold mos transistors," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, pp. 607–616, May 1999.
- [41] B. Gilbert, "Translinear circuits: a proposed classification," *Electronics Letters*, vol. 11, pp. 14–16, January 1975.
- [42] Y. Tsividis, *Operation and Modeling of the MOS Transistor*. Boston, MA: McGraw-Hill, 2002.
- [43] A. Andreou, "On physical models of neural computation and their analog vlsi implementation," in *Proceedings of Workshop on Physics and Computation*, pp. 255–264, Nov 1994.
- [44] E. Vittoz, "Low-power design: ways to approach the limits," in *Proceedings of the International Solid State Circuits Conference*, pp. 14–18, Feb 1994.
- [45] I. Arel, D. Rose, and T. Karnowski, "Deep machine learning - a new frontier in artificial intelligence research [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 5, pp. 13–18, Nov 2010.
- [46] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron, *et al.*, "Theano: Deep learning on gpus with python," in *NIPS 2011, BigLearning Workshop, Granada, Spain*, 2011.
- [47] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1tops/w analog deep machine-learning engine with floating-gate storage in 0.13 μ m cmos," in *Proceedings of the International Solid State Circuits Conference*, pp. 504–505, Feb 2014.
- [48] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. on Systems, Man and Cybernetics- part B*, vol. 42, no. 2, pp. 515–29, 2012.
- [49] M. Shoaib, N. Jha, and N. Verma, "Enabling Advanced Inference on Sensor Nodes through direct use of Compressively Sensed Signals," in *Proc. of DATE*, pp. 437–442, 2012.
- [50] S. Zhak, M. Baker, and R. Sarpeshkar, "A Low-Power Wide Dynamic Range Envelope Detector," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 1750–53, Oct 2003.
- [51] T. Delbruck and A. V. Schaik, "Bias current generators with wide dynamic range," *Analog Integrated Circuits and Signal Processing*, vol. 43, no. 3, pp. 247–68, 2005.
- [52] A. Kamboh and A. J. Mason, "Computationally Efficient Neural Feature Extraction for Spike Sorting in Implantable High-Density Recording Systems," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 21, no. 1, pp. 1–9, 2013.
- [53] S. Gibson, J. Judy, and D. Markovic, "Comparison of spike-sorting algorithms for future hardware implementation," in *Proceedings of the International IEEE EMBS Conference on Neural Engineering*, pp. 5015–5020, Aug 2008.

- [54] A. Kamboh and A. Mason, “On-chip feature extraction for spike sorting in high density implantable neural recording systems,” in *Proceedings of the IEEE Biomedical Circuits and Systems Conference*, pp. 13–16, Nov. 2010.
- [55] G. Buzsaky and K. Mizuseki, “The log-dynamic brain: how skewed distributions affect network operations,” *Nature Reviews Neuroscience*, vol. 15, pp. 264–278, 2014.
- [56] K. Awais and M. Andrew, “On-chip feature extraction for spike sorting in high density implantable neural recording systems,” in *Proceedings of the IEEE Biomedical Circuits and Systems Conference*, pp. 13–16, Nov 2010.
- [57] A. Lapolli, B. Coppa, and R. Heliot, “Low-power hardware for neural spike compression in bmis,” in *Proceedings of the International IEEE EMBS Conference on Neural Engineering*, pp. 2156–2159, July 2013.
- [58] N. Lotze and Y. Manoli, “A 62 mv 0.13 μ m cmos standard-cell-based design technique using schmitt-trigger logic,” *IEEE Journal of Solid-State Circuits*, vol. 47, pp. 47–60, Jan 2012.
- [59] M. Abeles and M. H. Goldstein Jr, “Multispike train analysis,” *Proceedings of the IEEE*, vol. 65, no. 5, pp. 762–773, 1977.
- [60] J. C. Letelier and P. P. Weber, “Spike sorting based on discrete wavelet transform coefficients,” *Journal of neuroscience methods*, vol. 101, no. 2, pp. 93–106, 2000.
- [61] Y. Yuan, C. Yang, and J. Si, “The m-sorter: an automatic and robust spike detection and classification system,” *Journal of neuroscience methods*, vol. 210, no. 2, pp. 281–290, 2012.
- [62] P.-M. Zhang, J.-Y. Wu, Y. Zhou, P.-J. Liang, and J.-Q. Yuan, “Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem,” *Journal of neuroscience methods*, vol. 135, no. 1, pp. 55–65, 2004.
- [63] J. Kao, S. Stavisky, D. Sussillo, P. Nuyujukian, and S. K., “Information systems opportunities in brain-machine interface decoders,” *Proceedings of the IEEE*, vol. 102, pp. 666–682, May 2014.
- [64] Y. Chen, E. Yao, and A. Basu, “A 128 channel 290 GMACs/W machine learning based co-processor for intention decoding in brain machine interfaces,” in *Proceedings of the International Symposium on Circuits and Systems*, pp. 3004–3007, May 2015.
- [65] E. Yao, Y. Chen, and A. Basu, “A 0.7 v, 40 nw compact, current-mode neural spike detector in 65 nm cmos,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2015.
- [66] S. Paraskevopoulou, D. Barsakcioglu, M. Saberi, A. Eftekhar, and C. T.G., “Feature extraction using first and second derivative extrema (fsde) for realtime and hardware-efficient spike sorting,” *Journal of Neuroscience Methods*, vol. 215, no. 1, pp. 29–37, 2013.

- [67] F. Corradi, D. Bontrager, and G. Indiveri, "Toward neuromorphic intelligent brain-machine interfaces: An event-based neural recording and processing system," in *Proceedings of the IEEE Biomedical Circuits and Systems Conference*, pp. 584–7, Nov 2014.
- [68] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 1212–24, June 2005.
- [69] B. Razavi, *Design of Analog CMOS Integrated Circuits*. Mc-Graw Hill Education, Aug 2000.
- [70] E. Neftci and G. Indiveri, "A device mismatch compensation method for VLSI neural networks," *IEEE Biomedical Circuits and Systems*, pp. 262–265, 2010.
- [71] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses With Spike-Timing Dependent Plasticity," *IEEE Transactions on Neural Networks*, vol. 17, pp. 211–221, Jan. 2006.
- [72] J. Arthur and K. Boahen, "Synchrony in Silicon: The Gamma Rhythm," *IEEE Transactions on Neural Networks*, vol. 18, pp. 1815–1825, Nov. 2007.
- [73] A. Basu and P. Hasler, "Nullcline based Design of a Silicon Neuron," *IEEE Transactions on Circuits and Systems I*, vol. 57, pp. 2938–47, Nov. 2010.
- [74] B. Linares-Barranco, T. Serrano-Gotarredona, and R. Serrano-Gotarredona, "Compact low-power calibration mini-DACs for neural massive arrays with programmable weights," vol. 14, pp. 1207–16, Sept 2003.
- [75] S. Brink, S. Nease, and P. H. et. al., "A Learning-enabled Neuron Array IC Based upon Transistor Channel Models of Biological Phenomenon," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, pp. 71–81, Feb. 2012.
- [76] S. Shuo and A. Basu, "Analysis and reduction of mismatch in silicon neurons," in *IEEE Biomedical Circuits and Systems*, (San-Diego, USA), Oct 2011.
- [77] T. Pfeil, A. Scherzer, J. Schemmel, and K. Meier, "Neuromorphic learning towards nano second precision," in *Proceedings of the International Joint Conference on Neural Networks*, (Dallas, USA), pp. 1–5, 2013.
- [78] J. Oh, G. Kim, B.-G. Nam, and H.-J. Yoo, "A 57 mw 12.5 μ j/epoch embedded mixed-mode neuro-fuzzy processor for mobile real-time object recognition," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 2894–2907, Nov 2013.
- [79] C. E. et. al., "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–05, 2012.
- [80] J. T. et. al., "Synthesis of neural networks for spatio-temporal spike pattern recognition and processing," *Frontiers in Neuroscience*, vol. 7, 2013.
- [81] A. V. Schaik and J. Tapson, "Online and adaptive pseudoinverse solutions for ELM weights," *Neurocomputing*, vol. 149, pp. 233–8, 2015.

- [82] A. Basu, S. Shuo, H. Zhou, M. H. Lim, and G. B. Huang, "Silicon Spiking Neurons for Hardware Implementation of Extreme Learning Machines," *Neurocomputing*, vol. 102, pp. 125–34, 2012.
- [83] Y. Enyi, S. Hussain, A. Basu, and G. B. Huang, "Computation using Mismatch: Neuromorphic Extreme Learning Machines," in *Proceedings of the IEEE Biomedical Circuits and Systems Conference*, Oct 2013.
- [84] U. M. L. repository, "<http://archive.ics.uci.edu/ml/>."
- [85] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme Learning Machines: A Survey," *Int. J. Mach. Learn. & Cyber.*, vol. 2, pp. 107–122, 2011.
- [86] G.-B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.
- [87] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, pp. 55–67, Feb. 1970.
- [88] G. I. et. al., "Neuromorphic Silicon Neuron Circuits," *Frontiers in Neuroscience*, vol. 5, May 2011.
- [89] R. Sarpeshkar, T. Delbruck, and C. Mead, "White noise in MOS transistors and resistors," *IEEE Transactions on Electron Devices*, vol. 9, pp. 23–29, Nov 1993.
- [90] B. Marr, B. Degnan, P. E. Hasler, and D. Anderson, "Scaling Energy Per Operation via an Asynchronous Pipeline," *IEEE Transactions on VLSI*, vol. 21, pp. 147–151, Jan. 2013.
- [91] Y. He and C. H. Chang, "A New Redundant Binary Booth Encoding for Fast 2n-Bit Multiplier Design," *IEEE Transactions on Circuits and Systems I*, vol. 56, pp. 1192–1201, June 2009.
- [92] M. L. G. de Solaz and R. Conway, "Razor Based Programmable Truncated Multiply and Accumulate, Energy-Reduction for Efficient Digital Signal Processing," *IEEE Transactions on VLSI*, vol. 23, pp. 189–93, Jan 2015.
- [93] C. S. Thakur, T. J. Hamilton, R. Wang, J. Tapson, and A. V. Schaik, "A neuromorphic hardware framework based on population coding," in *Proceedings of the International Joint Conference on Neural Networks*, (Ireland), July 2015.
- [94] A. Patil, S. Shen, E. Yao, and A. Basu, "Random projection for spike sorting: Decoding neural signals the neural network way," in *Proceedings of the IEEE Biomedical Circuits and Systems Conference*, pp. 1–4, Oct 2015.
- [95] Y. Chen, E. Yao, and A. Basu, "A 128-channel extreme learning machine-based neural decoder for brain machine interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, pp. 679–692, June 2016.
- [96] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. Chandrakasan, "A Micro-Power EEG Acquisition SoC With Integrated Feature Extraction Processor for a Chronic Seizure Detection System," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 804–16, April 2010.