A SYSTEM DESIGN APPROACH TO NEUROMORPHIC CLASSIFIERS

A Thesis
Presented to
The Academic Faculty

by

Shubha Ramakrishnan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2013

A SYSTEM DESIGN APPROACH TO NEUROMORPHIC CLASSIFIERS

Approved by:

Professor Jennifer Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Robert Butera
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Eugenio Culurciello
School of Biomedical Engineering
Purdue University

Professor Sung-kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: Jan 2013

ACKNOWLEDGEMENTS

I thank my advisor, Jennifer Hasler, for all her support, encouragement, and mentorship. I am deeply grateful to her for introducing me to several exciting areas of research, as well as allowing me freedom to pursue some of my own interests.

I thank my committee members, for the time they have taken to serve on my committee and for their feedback. I am especially thankful to David Anderson for his insight and guidance on noise-suppression and auditory processing.

I met several bright and wonderful people at the ICELAB, whom I enjoyed working with and who taught me a lot. I thank all the members of ICELAB for their help and camaraderie.

I thank my parents for their unconditional love, support, encouragement, and patience. I am grateful to my sister Prabha, for all the phone counseling. I am thankful to my parents-in-law for all their support.

My journey would have been impossible without my husband Rishi, who was with me every step of the way and also made it enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In our attempts to make our devices smarter, smaller, and increase battery lives, we need to consider alternative strategies to mainstream approaches to problem solving - namely analog and neuromorphic solutions for signal processing. Contrary to expectations from Moore's Law predicting an efficiency increase with technology node scaling, the efficiencies of modern digital processors have asymptotically approached 10 Giga Computations per Joule. It is clear that the next generation application need demands a much higher computing efficiency than is possible with following current trends. We need to investigate other techniques to increase computing efficiencies. The two strategies that we deal with in this research are analog and neuromorphic engineering. Carver Mead's hypothesis states that computation using the inherent physics of the device is $1000X$ more efficient than digital. It has already been shown that for certain functions such as Matrix Multiplication, Frequency decomposition, Analog FFT, Adaptive filtering, Winner Take All, analog is more efficient than digital processing. However, for analog signal processing to be a serious contender to digital techniques, it is essential that the hardware is programmable. Custom analog IC fabrication, although more efficient, is very expensive. Custom analog is also inflexible, making it difficult to make modifications to the processing algorithm or use it for slightly different applications.

Neuromorphic engineering deals with the design of systems that are either inspired by neuro-biological functions or replicate them. However, for real impact, it is essential to demonstrate significant advantages to main stream approaches in engineering applications by introducing a new paradigm of computing, new algorithms for information processing ultimately leading to more intelligent systems. This research

deals with developing emulation tools in silicon that enable investigation of neural computing principles and their pertinence to information processing in engineering applications.

Ultimately, this work aims to develop a neuromorphic system design flow for implementing speech classifier solutions, thereby demonstrating significant computational efficiency advantages over other existing implementations. This work uses low-power analog solutions for frequency analysis and feature extraction, and a neuromorphic approach to classification tasks. In doing so, this work also aspires to posit the significance of dendrites in neural computation.

CHAPTER I

NEUROMORPHIC DESIGN FOR ENGINEERING

A large portion of the research in electrical and computer engineering today is directed towards efficient computing to make smart and low power devices. In our attempts to make our devices smarter, smaller, and increase battery lives, we need to consider alternative strategies to mainstream approaches to problem solving - namely analog and neuromorphic solutions for signal processing. The reason for alternate approaches is due to the barrier to computing efficiency observed in today's digital computers. Marr et al published a survey of digital processors currently in the market by plotting their technology node and their computing efficiency [69]. These were in-market ICs and contrary to expectations from Moore's Law predicting an efficiency increase with technology node scaling, the efficiencies asymptotically approached 10 Giga Computations per Joule. The unit of computation here was considered to be a Million Multiply Accumulate (MMAC) operations, which is a fair metric for comparing DSPs, microprocessors as well as analog processors. We prefer to use the metric of MMAC for computation as opposed to Operations (Ops), since the MMAC actually defines a computation, while an Op could well be a no-Op. The survey from [69] is plotted in Fig. 1. A possible cause for the efficiency wall is the sub-threshold mismatch in devices that worsens as the technology scales. Designers have to spend more power and area compensating for the mismatch. It is clear that the next generation application need for smart devices demands a much higher computing efficiency than is possible with following current trends. We need to investigate other techniques to increase computing efficiencies. The two strategies that we deal with in this research are analog and neuromorphic engineering.

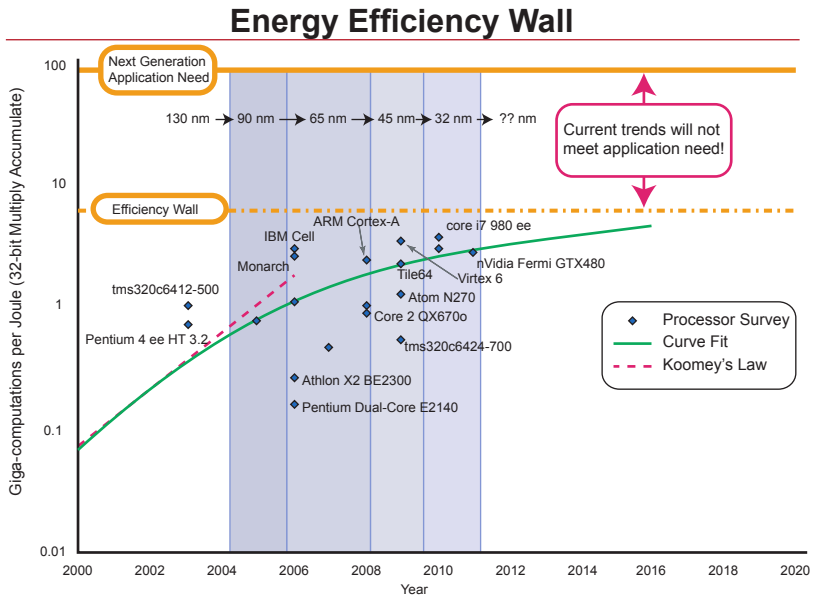


Figure 1: **Efficiency Wall for Digital Processors:** A barrier to the observed computing efficiency at 10 Giga Computations / Joule is plotted. Technology node scaling does not change the barrier. Sub-threshold mismatch is hypothesized as a reason for this efficiency barrier.

1.1 Analog Signal Processing

Carver Mead's hypothesis states that computation using the inherent physics of the device is $1000X$ more efficient than digital [72]. It has already been shown that for certain functions such as Matrix Multiplication, Frequency decomposition, Analog FFT, Adaptive filtering, Winner Take All, analog is more efficient than digital processing [17, 31]. However, for analog signal processing to be a serious contender to digital techniques, it is essential that the hardware is programmable. Custom analog IC fabrication, although more efficient, is very expensive. Custom analog is also inflexible, making it difficult to make modifications to the processing algorithm or use it for slightly different applications.

Programmable analog processing in the case of a multiplier is more efficient due to the computing element also having memory, as shown in Fig. 2. The local weight storage obviates the need for memory access, as in the case of a digital computing flow, thereby reducing power consumed during computation. Programmable analog also

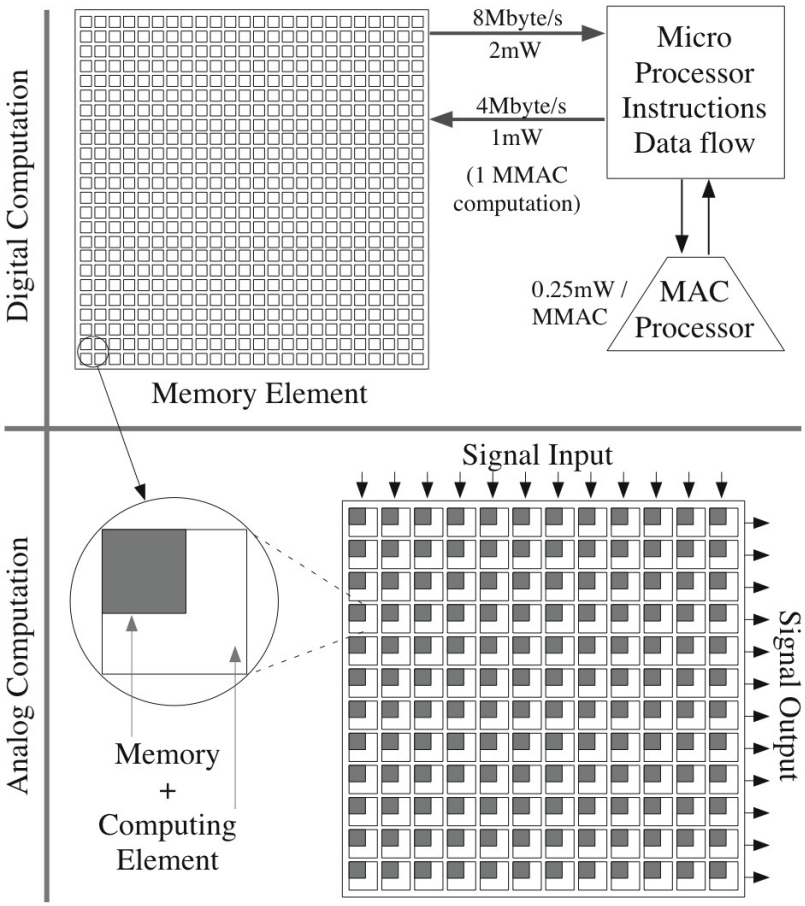


Figure 2: **Comparison of memory access costs:** In programmable analog, the computing and storage elements are located close to each other, while in digital they are separated. The cost of computing includes access to the memory.

has the great advantage of canceling mismatch effects using floating-gates. Hence, during the course of this work, we explore programmable analog strategies for signal processing.

1.2 Neuromorphic Engineering

Neuromorphic engineering deals with the design of systems that are either inspired by neuro-biological functions or replicate them. The goal of neuromorphic engineering may be developing system emulation tools for neuroscientists for investigative purposes with a significant improvement over fully digital solutions in terms of computing efficiency and size. Systems that closely model biology, with power supplies

similar to biology can be used for neural interfacing and stimulation in medical applications. Other goals include high-level modeling of signal processing pathways based on the structure of biological systems for improving performance/efficiency and detailed modeling of neural networks to exploit low-level processing for engineering applications.

In the two decades of its existence, progress has been made in bringing the fundamental principles of neuromorphic engineering to fruition in products like touch-pads and imagers, system designs such as silicon retina/imager, silicon cochlea. However, for greater impact, it is essential to demonstrate significant advantages to main stream approaches in engineering applications by introducing a new paradigm of computing, new algorithms for information processing ultimately leading to more intelligent systems. This research deals with developing emulation tools in silicon that enable investigation of neural computing principles and their pertinence to information processing in engineering applications.

Humans perform much better than current systems at speech recognition tasks. Low-power real-time compact implementations have several wide ranging applications such as in aids for the hearing impaired, search of speech databases, and enhancing human computer interactions. However, the best existing speech recognizers perform poorly on unconstrained speech, and do better with specific talkers when ample training data is available. Among the popular speech recognition algorithms in use today is one that uses Hidden Markov Model (HMM) techniques, but is also the most computationally challenging. However, it consistently performs better than other classifier architectures in speech recognition tasks which make it an attractive option to pursue. The high computation and memory requirements of the algorithm mandate massively parallel designs for real-time, large-vocabulary speech recognizers which raises the barrier for hardware implementation using conventional digital hardware approaches.

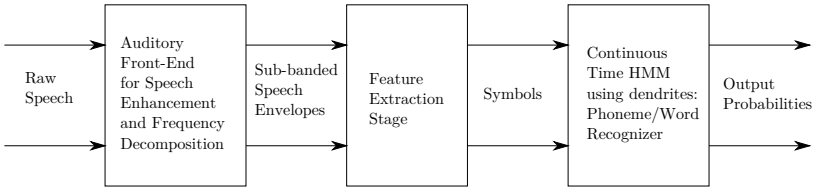


Figure 3: System flow for building neuromorphic classifiers.

This work aims to develop a neuromorphic system design flow for implementing HMM-based speech classifier solutions, thereby demonstrating significant computational efficiency advantages over other existing implementations. This work uses low-power analog solutions for frequency analysis and feature extraction, and a neuromorphic approach to classification tasks. In doing so, this work also aspires to posit the significance of dendrites in neural computation, which is often ignored by a significant portion of the neuroscience community.

The system shown in Fig. 3, includes a sensory front-end built based on high-level biological modeling of the cochlea to enhance speech and extract sub-band signal energies. These inputs are transformed into "symbols" or "parts of speech" representation using a weighted summation, after which the probability that a "symbol" is detected is estimated. These input probabilities are supplied to a continuous-time HMM, which computes the output probabilities for phonemes/words.

This document is organized as follows. In Chapter 2, we describe architectures for audio classification front-ends on a reconfigurable analog platform. Real-time implementation of audio processing algorithms involving discrete-time signals tend to be power-intensive. We present an alternate continuous-time system implementation of a noise-suppression algorithm on our reconfigurable chip, while detailing the design considerations. We also describe a framework which enables implementations of other speech processing algorithms, classifier front-ends and hearing aids.

In Chapter 3, a novel classifier structure is presented as a general-purpose, low-power, compact, programmable classifier architecture that is capable of greater computation than a 1-layer neural network, and equivalent to a 2-layer perceptron. The classifier generates event outputs and is suitable for integration with event-driven systems. The main sources of mismatch, temperature dependence and methods for compensation are discussed. We present measured data from simple linear and non-linear classifier structures and analyze the power and computing efficiency for scaled structures.

In Chapter 4, a novel neuromorphic chip that models neurons for efficient computation is discussed. Traditional architectures of neuron array chips consist of large scale systems that are interfaced with AER for implementing intra- or inter-chip connectivity. We present a chip that uses AER for inter-chip communication but uses fast, reconfigurable FPGA-style routing with local memory for intra-chip connectivity. We model neurons with biologically realistic channel models, synapses and dendrites. This chip is suitable for small-scale network simulations and can also be used for sequence detection, utilizing directional selectivity properties of dendrites, ultimately for use in word recognition.

In Chapter 5, we describe a single transistor floating gate synapse device that can be used to store a weight in a non-volatile manner, compute a biological EPSP, and demonstrate biological learning rules such as LTP, LTD and STDP. We also describe a highly scalable architecture of a matrix of synapses to implement the described learning rules. Parameters for weight update in the 0.35μm process have been extracted and can be used to predict the change in weight based on time difference between pre- and post-synaptic spike times.

Finally, in Chapter 6, we describe a new IC that integrates all the elements required for building low power, efficient neuromorphic classifiers. We list the choices made in the design process and describe the functional blocks.

CHAPTER II

SPEECH PROCESSING ON A RECONFIGURABLE ANALOG PLATFORM

We present a reconfigurable analog chip that can be used as a front-end for audio processing and signal enhancement. The possible benefits of analog in terms of power dissipation per unit computation has long been hypothesized by Mead in [72]. A careful study of the same done in [92] showed that for a particular system, analog processing would be better than digital if the desired signal to noise ratio (SNR) was below a certain value. Since then, there have been several custom analog implementations of various signal processing blocks exploiting this feature [19, 20, 57, 80, 95]. The popularity of analog processing, however has remained far lesser than digital owing to difficulty in design and fixed functionality. However, recent developments in field programmable analog arrays (FPAAs), the analog equivalent of FPGAs, shows great promise in allowing the end user to easily utilize the power of analog processing. The scope of such a structure in performing signal processing was discussed earlier in [108]. FPAAs offer the advantage of rapid prototyping and programmability, allowing the user to implement a wide variety of circuits, unlike expensive custom analog IC fabrication. In this work, we use the FPAA to implement a few signal processing algorithms that are useful in audio processing. We introduce the reader to the trade-offs in implementing a few algorithms on a reconfigurable platform and demonstrate a few examples of noise suppression algorithms based on a physiological model of hearing that use non-linear filtering in different sub-bands. The motivation and approach to audio processing adopted in this work, shown in Fig. 4 is closely

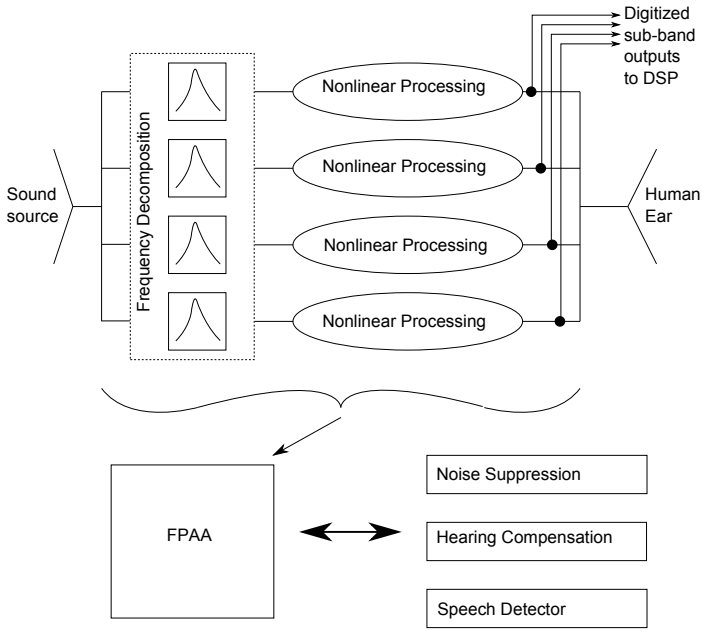


Figure 4: **High level overview:** We envision a range of biologically inspired signal processing algorithms, that fit into the pathway between speech production (source) and perception (human ear). These algorithms are implemented by non-linear processing of sub-banded speech signals for applications such as noise suppression or hearing compensation, by proper choice of the non-linearity. In addition, the outputs of the non-linear processor can be taken at each sub-band, for speech detection instead of recombining to generate a perceptible signal for the human ear.

related to the model for the human auditory system [62]. The frequency decomposition performed by the basilar membrane is modeled using a bank of parallel bandpass filters, with exponentially spaced center frequencies. The inner and outer hair cells, which detect the sound intensity and provide non-linear amplification respectively, are modeled by the non-linear signal processing block. Using this general framework, a variety of non-linear processing can result in applications ranging from noise suppression systems, speech activity detectors, speech classifiers and hearing aid blocks. The signal is recombined post-processing and converted back into an audio signal, or digitized before recombination, providing sub-banded input for further DSP. This structure is amenable to easy implementation on the FPAA, making it an attractive platform for comparing various analog signal processing (ASP) algorithms for audio applications. This chapter is organized as follows: Section II examines a few gain

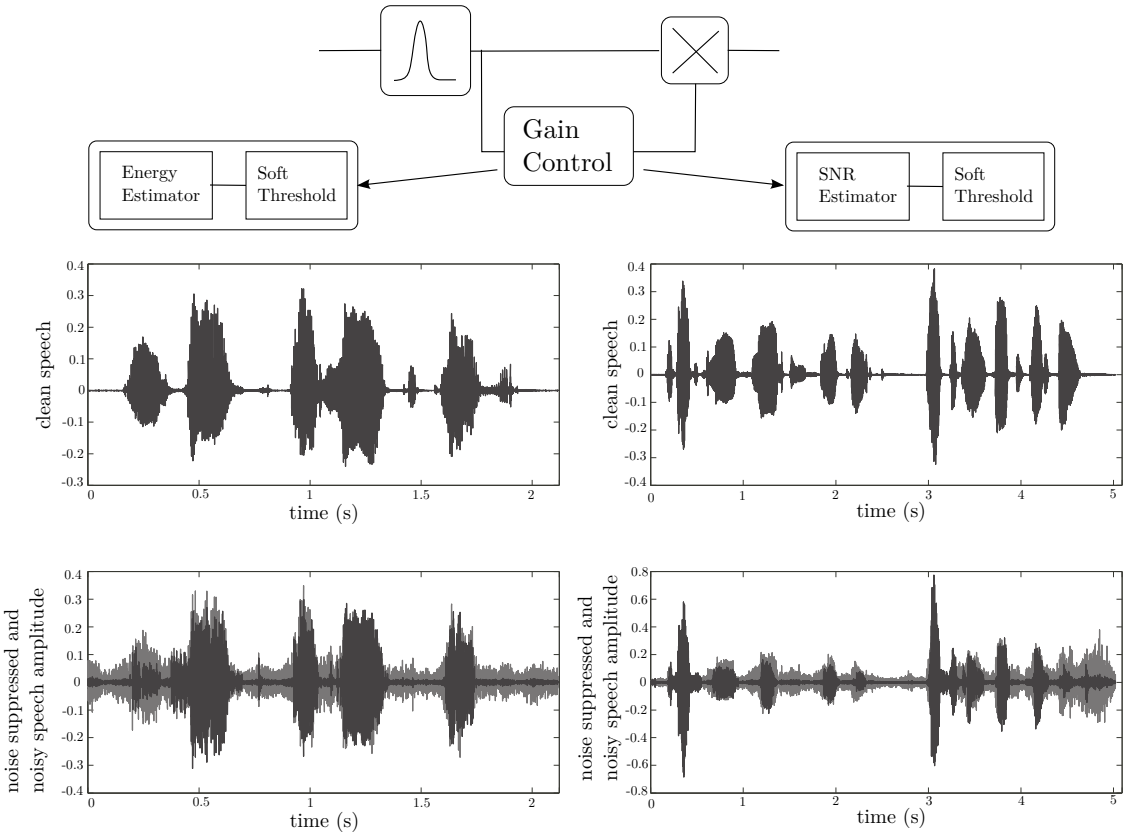


Figure 5: **System Overview:** Top level signal flow consists of frequency decomposition of incoming speech signal into exponentially spaced sub-bands. The non-linear processing consists of signal energy detector and a gain control block which modulates the gain applied to a signal-signal multiplier block. The outputs from the multipliers are then recombined to produce the processed output. For a non-linear processing block consisting of a speech envelope detector and a thresholding block, results from a MATLAB simulation showing clean, noisy(gray) speech with 5 dB SNR and processed speech. MATLAB simulation results for a processing block consisting of an SNR estimator and a threshold block are also shown.

control topologies and their applicability to noise suppression algorithms. Section III provides a short review of the reconfigurable hardware that is being used. Section IV and V discuss circuits and measurement results of individual blocks used in our algorithm and measurement results from a single channel implementing a processing algorithm respectively. In Section VI we present measured results for a few system implementations of noise suppression.

2.1 Speech Enhancement

Today, most efforts in audio signal enhancement have been concentrated in processing the digitized signal with Wiener filtering, spectral subtraction and other techniques [58, 90]. We consider an approach in which the signal is "enhanced" prior to/instead of digitizing, using the non-linear signal processing blocks available in the FPAA, to propose a solution for speech enhancement. Our solution follows the algorithm previously described in [3, 21], suitable for implementation using analog VLSI. A noisy audio signal $x(t)$ can be represented as

$$x(t) = s(t) + n(t) \tag{1}$$

where $s(t)$ is the signal and $n(t)$ is the noise. We assume the noise is stationary over a longer period of time relative to the speech signal, resulting in a separation of timescales in $s(t)$ and $n(t)$. We estimate the noise $\hat{n}(t)$, from $x(t)$ and then modulate the gain of the signal in the following stage. When $x(t) > \hat{n}(t)$, our audio signal dominates our noise estimate. Hence, we apply a large gain to the signal, to emphasize speech portions of the signal. When $x(t) \approx \hat{n}(t)$, the audio signal is mostly noise and we reduce the signal gain, so that the noisy portions may be suppressed.

Fig. 5 depicts the approach taken in this research for noise suppression. The noisy audio signal is sub-banded using a second-order bandpass filter. We represent an acoustic signal as a sum of band-limited signals; each sub-band representation is further decomposed into a product of an envelope (which carries the instantaneous loudness information) and a rapidly oscillating signal (carrier) of nearly constant power. This signal representation can be applied to auditory analysis by making the signal sub-bands roughly equal in bandwidth to the critical bands in the ear. In particular, the acoustic signal s(t) can be expressed as

$$s(t) = \sum_i e_i(t) v_i(t) \tag{2}$$

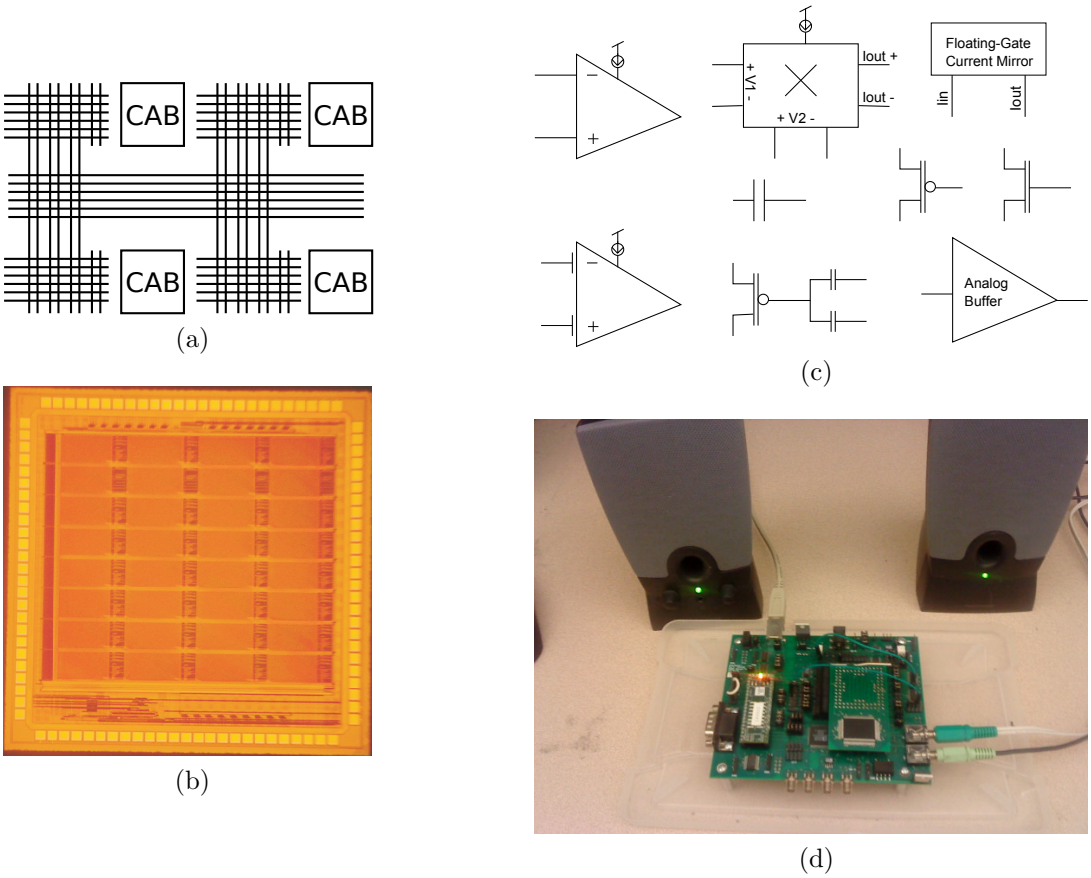


Figure 6: **Overview of RASP 2.8a chip:** (a) This FPAA consists of computational analog blocks(CABs) embedded in a routing fabric made up of programmable switches. (b) A die photo of the device fabricated in a 0.35μm CMOS process. (c) The CAB consists of commonly used analog functional blocks such as OTAs, signal-signal multipliers, transistors, MITEs and capacitors. (d) Our test infrastructure includes a PCB that uses USB for power and communication. The board consists of a micro-controller, DACs, ADCs for programming and testing. The board also has audio ports and amplifiers that can be used to drive speakers.

where $v_i(t)$ is the speech excitation (high frequency) and $e_i(t)$ is the speech envelope (low frequency) in the i^{th} channel. The band-limited signal envelope is estimated using an envelope detector, followed by a gain control block which modulates the gain of the signal provided by a signal-by-signal multiplier.

2.1.1 Known signal quality

The gain control block can be simplified with a priori knowledge of the incoming speech signal quality. For signals with reasonably high SNR, we can assume that

when the sub-banded noisy speech envelope falls below a certain value, there is no actual speech signal and it is mostly noise. A soft thresholding function applied to the envelope estimate then results in attenuation of all signals with an envelope below a certain value and a gain applied to all signals above the threshold. Fig. 5 shows the results from a MATLAB simulation for such a block. This technique results in loss of stand-alone soft sounds in the speech. Furthermore, this algorithm is limited in the sense that it performs poorly on inputs with low SNR.

2.1.2 Unknown signal quality

This method does not make assumptions about the input SNR, but assumes that the temporal characteristics of background noise and speech signal are different [22]. Using that information, we can arrive at a noise estimate based on the noisy input speech. Dividing the noisy speech envelope by the noise estimate gives us a measure of the "instantaneous SNR". A soft thresholding function is now applied to the instantaneous SNR estimate, to emphasize portions of the input with moderate SNR and attenuate portions with low SNR. Fig. 5 plots the simulated results for such an algorithm. This method performed better in retaining soft sounds, unless the masking noise was present in the same band as the soft sound. We also expect that this technique performs better with low SNR inputs. Simulations indicate that a soft-thresholding function is essential to avoid artifacts of processing such as harsh clipping sounds generated due to the switching of the gain-control block output. In this work, we do not consider the non-linearities generated due to soft-thresholding.

2.2 Reconfigurable Analog Hardware

The chip used in our implementation has been described previously [6]. Nevertheless, we briefly describe it here for the sake of completeness. The Rasp 2.8a has 32 Computational Analog Blocks (CABs) embedded in programmable routing fabric, as shown in Fig. 6a. The CABs consist of commonly used analog circuit components such

as Operational Transconductance Amplifiers (OTA), signal multipliers, floating capacitors, voltage buffers, multiple-input floating-gate (FG) transistors, transmission gates and transistor arrays. Some of the OTAs have FG inputs to the differential pair (FGOTA) due to the presence of capacitive dividers for increasing input linear range. All of the OTAs, buffers and multipliers are biased using precisely programmable FG transistors giving the user the flexibility to make trade-offs between bandwidth and power consumption. The range of the FG programming is 6 pA to 20 μA, with an accuracy of 9.5 floating-point bits [6]. FGs demonstrate excellent charge retention properties, with a charge loss of $< 1\%$ over a period of 10 years [104]. The interconnection between the various CAB components is made using the routing fabric which are of various types, allowing the user to select global routing lines for connecting blocks that are spaced far apart, local lines for connecting blocks within the same CAB and nearest neighbor lines, allowing connection between neighboring CABs. These offer flexibility to the user to choose a routing scheme which minimizes parasitic load capacitance for sensitive analog nodes (by choosing local or nearest neighbor lines) or a low resistance path for signals that travel long distances (by choosing global routing lines). Another advantage of this chip is that the interconnect switches are also FG transistors that can be programmed in an analog fashion and not just as ON/OFF switches [107]. Hence these can also be used as circuit elements, increasing the number of computational elements per unit area of the chip. The maximum signal bandwidth through the switch matrix is 57 MHz and the highest achievable filter bandwidth is 5 MHz, which makes it suitable for implementing audio processing algorithms. The test platform for using the FPAA is powered on USB which doubles up as a communication interface. It also has two sound ports for input and output stereo sounds, seen in Fig. 6d. The user can specify the desired signal processing function from a high level interface in SIMULINK, a software product by Mathworks.

An intermediate processing software is used which first converts the SIMULINK description to a SPICE netlist, which is then converted into target switch addresses on the chip [5], [82].

2.3 System Components

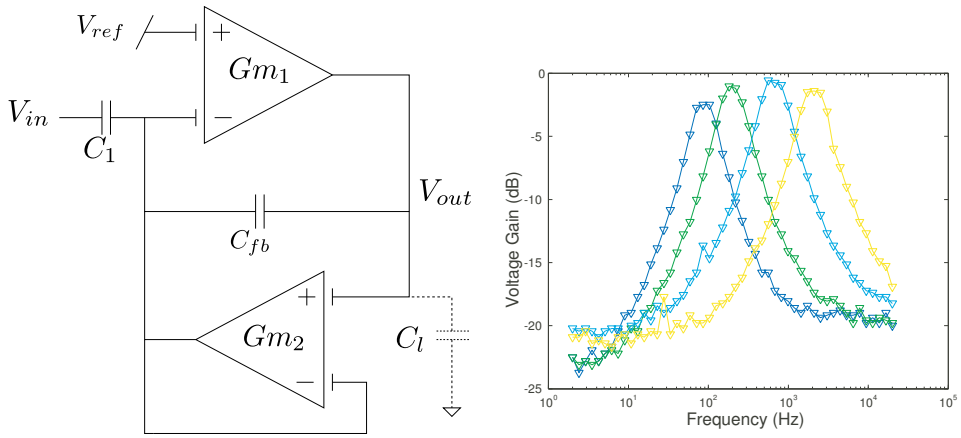
In this section, we discuss the various components used in the signal flow and present measured results from characterizing each individual block.

2.3.1 Band Pass filter

The first stage of the signal processing chain involves frequency decomposition. We use a bank of second-order bandpass filters that have exponentially spaced center frequencies. The basilar membrane can be modeled as a bank of bandpass filters with a 20 dB/decade roll off on the low frequency side, and a steeper roll off on the high frequency side. In practice, a filter with 20 dB/decade roll off is sufficient to separate the signal into bands. In a hearing aid model, we would choose the bandwidth of each filter to equal the bandwidth of the corresponding critical band in the cochlear model. The OTAs and FGOTAs in our chip enable implementation of Gm-C filters with tunable frequency responses. Their bias currents are set by FG devices which can be precisely programmed. The general expression for the subthreshold transconductance of the OTA used is

$$Gm = \kappa I_b / 2 * U_T \tag{3}$$

where I_b is the bias current, κ is the subthreshold slope of the transistors making up the differential input pair of the OTA and U_T is the thermal voltage. Fig. 7a depicts the schematic of a second-order bandpass filter compiled on the RASP2.8a, which is an OTA-based implementation of the circuit described in [103]. We use OTAs with FG differential inputs in this structure to improve the input linear range of the



(a) (b)

Figure 7: **Band Pass Filter:** (a) Schematic of the second order bandpass filter used for frequency decomposition. (b) Measured results from a bank of 4 bandpass filters with exponentially spaced center frequencies. The mismatch in the passband gain is due to the parasitic load capacitance C_L.

system. The transfer function of the filter is given by

$$\frac{V_o(s)}{V_i(s)} = \frac{\frac{C_1}{C_{FB}} s\frac{\tau}{Q}(s\tau_z - 1)}{s^2\tau^2 + s\frac{\tau}{Q} + 1} \tag{4}$$

where $\tau = \sqrt{\frac{C_1 C_{FB}}{Gm_1 Gm_2}}$ is the filter time constant, $Q = \sqrt{\frac{C_1}{C_{FB}}\frac{Gm_2}{Gm_1}}$ is the quality factor, and $\tau_z = \frac{C_{FB}}{Gm_1}$. Gm_1 and Gm_2 are the subthreshold transconductances of the forward and reverse amplifiers respectively. Intuitively, the filter is a second order section where the bias current of the forward amplifier sets the high-frequency cutoff and the bias current of the feedback amplifier sets the lower cutoff frequency. The midband gain is set by the ratio of the two capacitors. The additional filter zero can be placed at frequencies outside the speech band, thereby relying on low-pass filtering by the signal multiplier and the human ear itself to minimize its effects. The same filter structure is used but the bias currents are varied to get different center frequencies. Since the center frequency is given by $f_c = \frac{1}{2*\pi}\sqrt{\frac{Gm_1 Gm_2}{C_1 C_{FB}}}$ and $Gm \propto I_b$, the higher frequency channels dissipate more power than the lower frequency channels. Fig. 7b shows the measured frequency response of a bank of four filters compiled on our chip. The variability in gain is seen due to parasitic capacitance C_L at the output

node causing additional loading. The mid-band gain including the effect of parasitic loading at the output is given by

$$A_v = -\frac{C_1}{C_{FB}} \frac{1}{1 + \frac{Gm_2}{Gm_1}\frac{C_L}{C_{FB}}} \tag{5}$$

The effect of the parasitic capacitance can be lowered by increasing C_{FB}, but to maintain the same filter Q and center frequency, C_1, Gm_1 and Gm_2 will also have to be proportionally increased. Making a sensible routing choice for the output node helps in reducing the variability in gain. Since this node is sensitive to parasitic capacitance, it is preferable to use local or nearest-neighbor routing lines for making connections. Also, a voltage buffer is used at the output of the filter to prevent loading due to the following block.

2.3.2 Envelope Detector

To extract the envelope we take advantage of the non-linear behavior exhibited by a simple source follower. A pFET based source follower shown in Fig. 8a actually acts as a minimum detector. V_{out} quickly follows V_{in} with a gain of κ when the input decreases, but charges up at a rate $\frac{I_\tau}{C_L}$, where C_L is the capacitance at the output node. An nFET based source follower functions as a maximum-detector and can be used to extract the positive envelope of the signal. Using an OTA based peak detector, shown in Fig. 8b instead of a regular capacitively loaded source follower makes the circuit more sensitive to small changes in input. The gain now becomes $\frac{\kappa A}{(1+\kappa A)}$ where A is the open loop gain of the amplifier. The attack time constant of the minimum detector is also decreased by a factor of $(1 + \kappa A)$, allowing a faster response. Kirchoff's current law at the output node can be written as,

$$C_L \frac{dV_{out}}{dt} = I_o e^{(\kappa V_{in} - V_{out})/U_T} - I_\tau \tag{6}$$

We assume sub-V_T saturation operation for the input device and the transistor implementing the current source. Removing DC bias from (6) by using $I_o e^{(\kappa V_{in} - V_{out})/U_T} =$

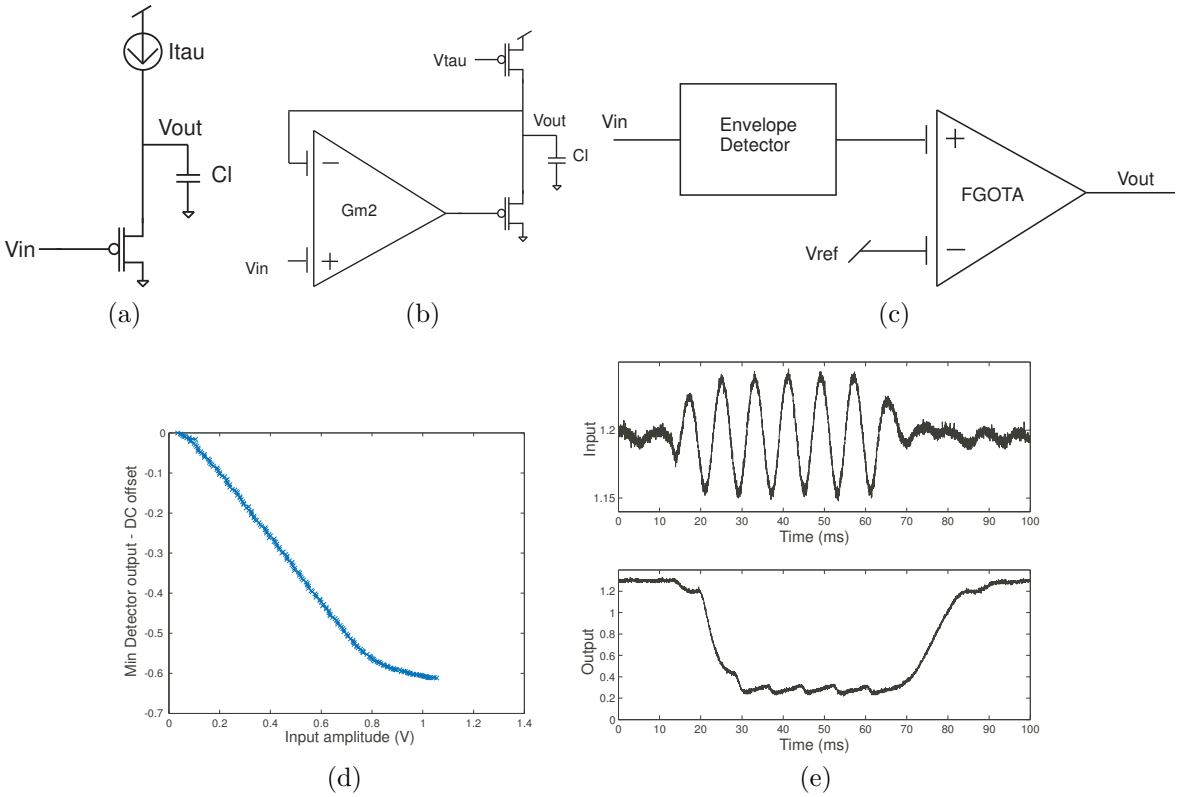


Figure 8: **Envelope Detector:** (a) and (b) Schematics of minimum detector circuits implemented on our chip. (c) Minimum detector followed by a FGOTA, providing a gain of about 5. (d) Measured transfer curve of the minimum detector showing output voltage versus input amplitude. (e) Transient output of the amplified envelope detector.

I_τ, we get

$$C_L \frac{dv_{out}}{dt} = I_\tau [e^{(\kappa v_{in} - v_{out})/U_T} - 1] \tag{7}$$

Using normalized variables $x = \frac{\kappa v_{in}}{U_T}$, $y = \frac{v_{out}}{U_T}$, and $\tau = \frac{C_L U_T}{I_\tau}$, the dynamics of the output voltage for the pFET based source follower can be written in normalized form as

$$\tau \frac{dy}{dt} = e^{x-y} - 1 \tag{8}$$

where I_τ is the quiescent bias current flowing through the circuit, set by V_τ. x and y refer to the normalized input and output voltages. For an OTA based minimum-detector, the dynamics remain the same, but $\tau = \frac{C_L U_T}{I_b(1+\kappa A)}$, resulting in a faster response time. The transfer function of the minimum detector is plotted in Fig. 8d.

The "'attack"' and "'release"' parameters of the envelope detector can be modified by changing the bias current. For the i^{th} channel, we choose the bias current for the peak detector such that the rate of decay of its output I_b/C_L does not cause a significant change in output voltage in a time period corresponding to the lowest frequency signal in that band. Choosing a very small bias current will cause the envelope detector to miss portions of the envelope, especially in the higher frequency signals within the i^{th} band. A higher than optimum bias current choice will result in capturing the excitation signal in addition to the envelope itself, which is not desirable. Thus, the bias current for the lower frequency bands is lower than that of the higher frequency bands. While in this implementation this parameter is set using a DAC voltage V_τ, it can be replaced with a FG switch element in the routing fabric, thereby storing the parameter V_τ on-chip. The measured results from the minimum-detect block are plotted in Fig. 8e. For an increased gain from the envelope detector, we use an open loop FGOTA to further amplify the signal, shown in Fig. 8c.

2.3.3 Multiplier

A fully differential signal by signal multiply block present in the RASP 2.8a is used to control the gain for each individual channel in the system. The multiplier is the well known gilbert cell structure, shown in Fig. 9a [28]. Stacking of multiple differential pairs leads to voltage head-room issues in the traditional Gilbert cell. In our structure, we avoid this by folding signal currents. The cascode biases are generated using [73]. V_1 and V_2 are the two differential inputs to the multiplier. The multiplier block produces differential output currents which can be converted to single ended voltage using a FG current mirror. Fig. 9c plots the measured output voltage against the differential input voltage V_1 for several fixed values of V_2.

The frequency response of the multiplier block is shown in Fig. 9d. The bandwidth of the multiplier can be increased by programming a larger bias current. In the figure,

I_b refers to the current in the bias generation circuits. All the inputs to the multiplier are FG transistors, which allow for precise offset cancellation. Since the outputs of the multiplier blocks are differential currents, signals from multiple channels can be added by tying the output nodes together. The summed current is then converted into voltage using a single FG current mirror. The FG transistors at the multiplier inputs and in the current mirror enable offset cancellation due to mismatch. Fig. 9a shows a piece of our system with 4 sub-banded channels and 4 gain stages provided by the gilbert multiplier. Fig. 9b plots the sum of the 4 channels at the output. In this case, there is a gain of 0.5 for each channel in the multiplier. The signal attenuation in the highest-frequency channel is higher than other channels since the bandwidth of the compiled system was around 2 KHz.

2.4 Single Channel System Results

We now proceed to discuss single band systems that include blocks described in the previous section. The non-linear signal processing block in Fig. 5 can be implemented in a variety of ways to produce processed speech for different applications. For noise suppression of signals with SNR $\geq$ 10dB, we apply a soft threshold to the sub-banded signal envelope to determine the gain for that channel, shown in Fig. 10a. The soft-threshold block is implemented using a comparator with very low gain. This is realized using an FGOTA programmed to a bias current of $1nA$. Fig. 10b shows the transient results for such a system for a single tone input.

A system that expands the dynamic range of the input signal can result in noise suppression [79]. One way to achieve this is by ensuring a power law relationship with an exponent > 1 between the input and output. The non-linear function can be implemented in current mode employing the trans-linear principle using MITE transistors [74]. The voltage output from the envelope detector is converted to current using an FGOTA block which has a linear range of about $600mV$. Fig. 11a depicts a

circuit with a squaring non-linearity. I_{scale} is set by the reference voltage V_{scale}. The output of the squaring circuit is converted back to voltage using a trans-impedance amplifier. Fig. 11c plots the relation between input and output amplitudes and shows a 12 dB improvement in dynamic range. The transient response of the system for a single tone input is plotted in Fig. 11b.

While an expansive non-linearity can be used for noise-suppression, a compressive non-linearity can be used in hearing aid applications [15]. Hearing loss is characterized by loss of inner hair cells that impair the ability to discern low intensity sounds. To compensate for the hearing loss, hearing aids typically compress the dynamic range by implementing a power law relationship with an exponent < 1. The squaring circuit can be easily converted into a square root circuit, by changing the configuration of the capacitors on the MITE transistors. While we do not show results from a speech processing system for hearing aid applications, we believe that the FPAA would be a good tool for prototyping algorithms.

While the circuits described previously may be suitable for enhancement of speech signals with moderately high SNR, they do not perform well for inputs with SNR ≤ 10 dB. In these cases, it is preferable to estimate the noise portion of the speech input before further processing. To do a real-time estimation of the noise floor, we detect the minimum of the noisy sub-band envelope [70]. The integration time constant of the minimum detector is chosen to be slower than the envelope detector for that channel. Essentially, our noise floor estimate is the minimum of the envelope signal in a time window that is set by the bias current of the minimum detector. We choose this time window to be large enough that the speech signal in that band is too fast to cause a significant change in the noise estimate. We rely on a separation of timescales in the noise signal and speech signal within a particular band. The voltage output from the envelope estimator and the noise estimator are then converted into currents, as shown in Fig. 13b. The MITE transistors can be configured to do current mode

multiplication and division using the circuit shown in Fig. 14b [74]. The output of the divider circuit is plotted in Fig. 14a. The relevant signals in the SNR estimator circuit is shown in Fig. 13a. In this case, we choose to use a minimum detector as the first envelope estimator and a slow maximum detector on the envelope to arrive at the noise estimate. Since the current to voltage converter provides an inversion, the divider voltage output is low for high SNR estimates and high for low SNR. The divider output is now thresholded using a comparator to produce the control voltage for the multiplier.

2.5 Multi-Channel System Results and Discussion

A four channel noise suppression system was implemented on the Rasp 2.8a chip. The maximum number of channels that can be implemented on this chip is 8, limited by the number of gilbert multiplier blocks. For the multi-band system, the time constants for the envelope detector and noise estimator blocks were set independently using on-board DACs. The envelope detector for the lowest frequency channel is tuned to have the highest time constant.

Transient output waveforms from the envelope-threshold and SNR estimator systems are plotted in Fig. 12. We used speech samples from the NOIZEUS database, which provides acoustically recombined noisy speech samples with fixed SNR [43]. Noisy speech with 0 dB and 5 dB SNR was played and the output from our system was recorded. Both systems resulted in substantial reduction of background noise present between speech. In the envelope-threshold system, the intelligibility of the output suffered, since this system resulted in harsher clipping sounds, cutting out soft sounds in the speech itself.

The SNR estimator system performed better with overall processed speech quality. Listening tests on the processed speech revealed that the system effectively reduced noise in between speech portions, but failed to do so during the speech. This behavior

is expected and is illustrated by Fig. 15. Channels 2 and 3 are active when speech is detected and the SNR is high enough. However, this allows the noise present in these channels to also leak through to the output. Increasing the number of bands will increase the ability of the system to resolve speech and noise into multiple bands, thereby allowing it to suppress the noise better. We conducted blind subjective hearing tests on a group of graduate students, the results of which are tabulated in 1. The students were asked to compare the noisy and processed speech on the background noise level and intelligibility and mark the sample that had lower noise and higher intelligibility. The percentages of people who preferred the processed speech in these metrics is presented in Table 1. A majority of students felt that the processed speech was more noise-suppressed than the original speech. However, the quality of our output rated lower than the original speech, which we attribute to the hardware limitation of 4 bands in the system. The spectrogram of the processed speech for selected inputs is plotted in Fig. 16, and shows that our system suppresses noise considerably for moderately low input SNR.

Table 1: Subjective evaluation of noise suppression algorithm on speech samples with added pink noise

SNR	Subjective (%): Noise level	Subjective (%): Intelligibility
0.7	100	35
2.6	100	40
4.5	100	40
5.5	90	0
10.6	90	0
13	80	5

2.5.1 Power consumption

Increasing the number of bands allows separation of the speech and noise activity, resulting in better noise suppression, but will also increase the power consumption.

The power consumed in individual blocks is listed in Table 2. The total power consumption for the 4 channel system, without including the FG programming circuitry and the amplifiers to drive the audio ports is $128.03\mu W$. A significant portion of this power is consumed in the buffers, that are necessary for driving sensitive analog signals over routing lines with large parasitic capacitance. The projected power consumption for a 32 channel system is $1.02mW$. The power consumption of all blocks except the bandpass filter can be linearly scaled. The filter power consumption is dependent on its center frequency, but since it is a small fraction of the total power, we assume it to be constant.

Table 2: Power consumption of individual blocks for a 4-channel system.

Functional Block	Power consumption
Bandpass filter	$0.4\mu W$
Noise estimator	$3.84\mu W$
Speech estimator	$4.03\mu W$
Multiplier	$5.52\mu W$
Divider and gain control	$27.84\mu W$
Buffers	$86.4\mu W$
Total	$128.03\mu W$

2.6 Conclusions

Our work describes how the reconfigurable chip (FPAA) described in [6], can be used for implementing signal processing algorithms, specifically for audio applications. The framework developed in this research also supports other applications such as voice-activity detection, hearing compensation, and classifier front-ends. The FPAA provides the user the flexibility in implementing circuits for analog signal processing while avoiding the costs of custom analog IC fabrication. Significant effort is spent in custom IC design to overcome mismatch effects. In our implementation, circuit parameters are set using FG biases that can be precisely programmed, enabling compensation of offsets due to component mismatch. This feature is exploited in

tuning the multiplier, bandpass filter and SNR estimator blocks. In the current implementation, the programmable biases show a temperature dependence. However, techniques for temperature compensation can be applied, where the gate coupling voltage to all FG transistors is supplied from a bootstrap reference. This ensures that the bias current (in weak inversion) is independent of temperature. One approach to noise suppression discussed in this work was previously published in [22]. However, we provide measured data for the first time from an integrated system compiled on a single reconfigurable chip. We also discuss three different approaches to noise suppression where we use implementations for the system components based on the blocks present in the FPAA, while describing trade-offs in performance. We acknowledge that there are better noise suppression algorithms in the literature, and the goal of this research is not to develop the best noise suppression system. We hope this chip makes analog signal processing more accessible to a wide audience.

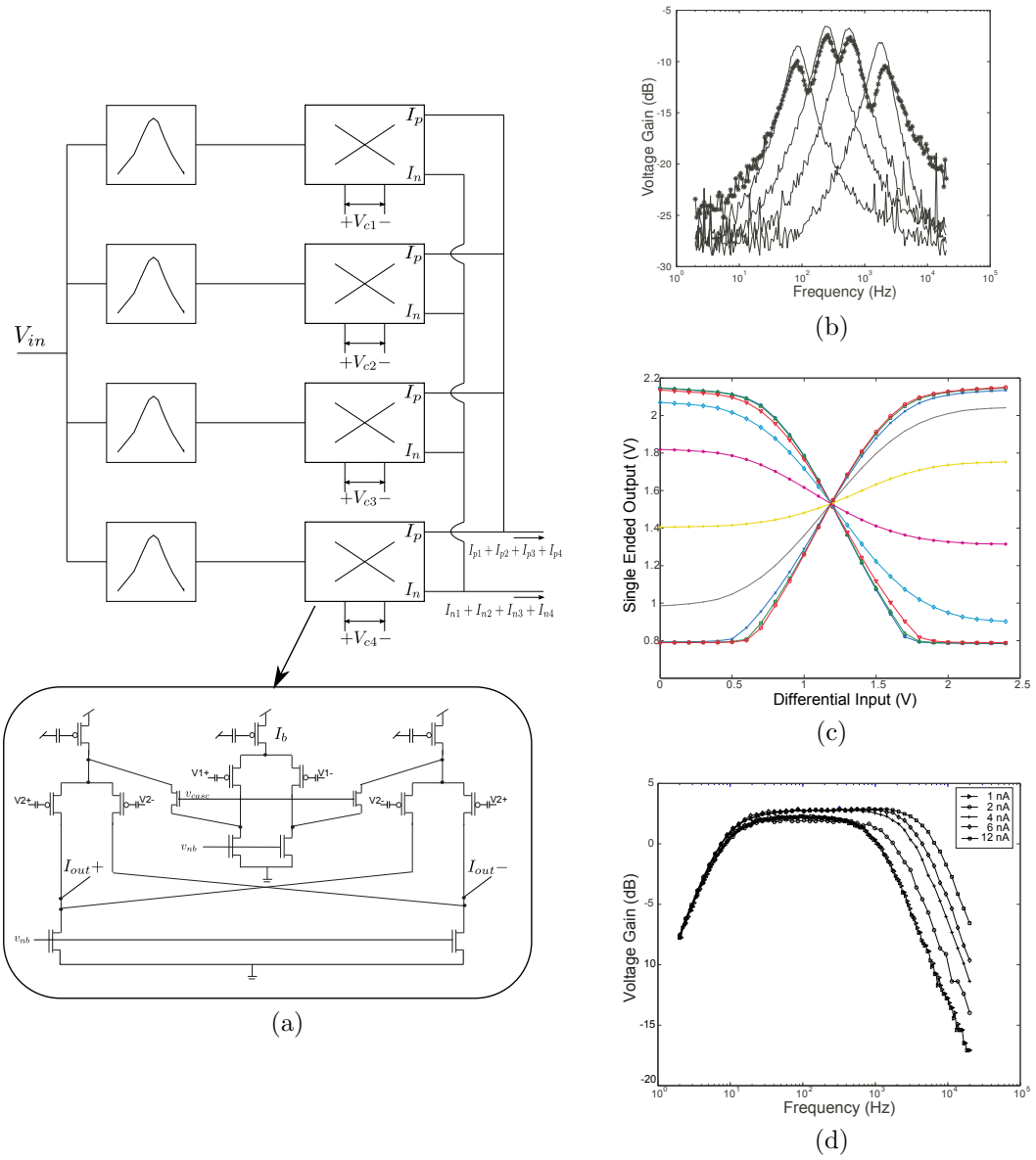
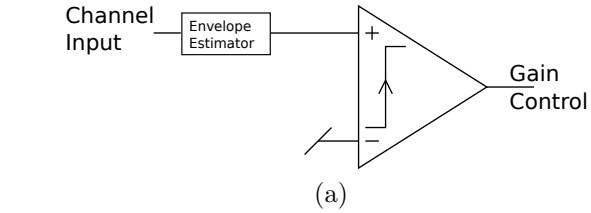
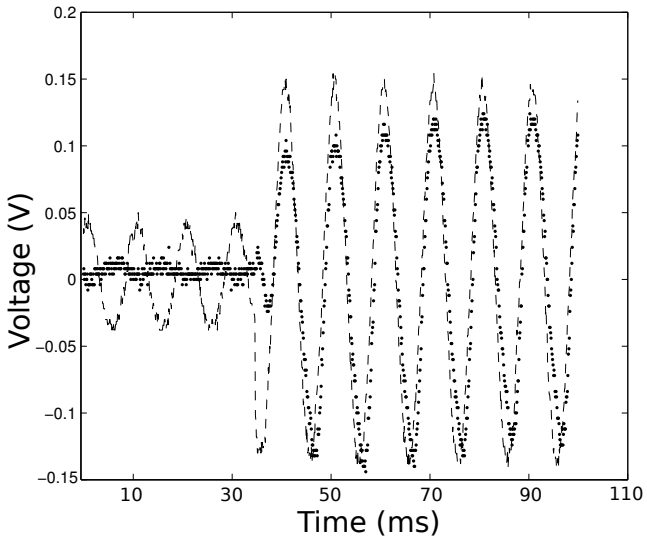


Figure 9: **Signal Multiplier:** (a) Feedforward system with four channels and four multipliers. The multipliers have current outputs that allow channel summing by tying together the outputs of individual channels. The multiplier has programmable bias current, and FG fully-differential inputs that allow offset cancellation. (b) Frequency response of the feedforward system, with summed responses from all four channels. The lines and dots denote the individual channels and the summed response, respectively. The gain of the fourth channel is lower than that of the other channels since the bandwidth of the system has been programmed to be lower than the center frequency of the fourth channel. (c) Multiplier DC response: V_{out} versus V_1 for different V_2. The current output of the multiplier is converted into voltage using a current mirror fashioned out of transistors in the CABs. (d) The frequency response of the multiplier. The corner frequency can be tuned by programming a higher bias current.

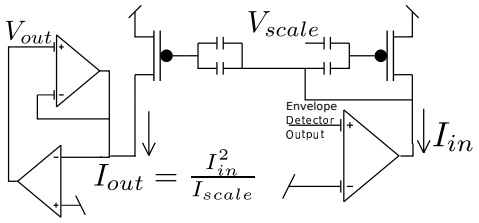


(a)

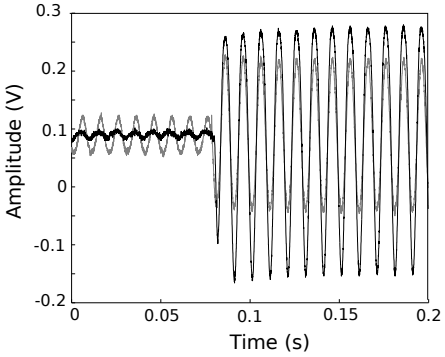


(b)

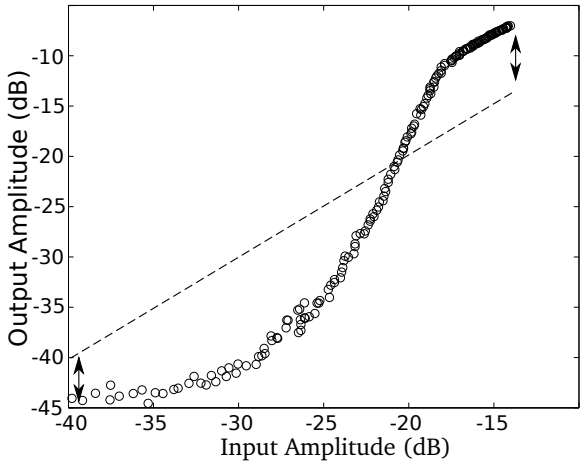
Figure 10: **Single Channel results for envelope thresholding system :** (a) Channel gain is determined by thresholding the envelope estimate. (b) Transient results for single tone inputs (Dashed trace is the input and Dotted trace is the output).



(a)

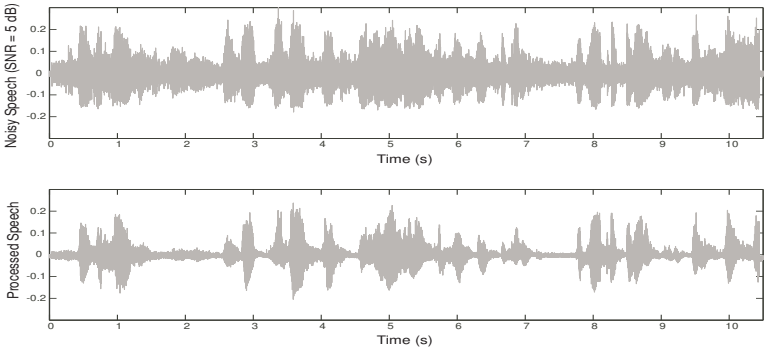


(b)

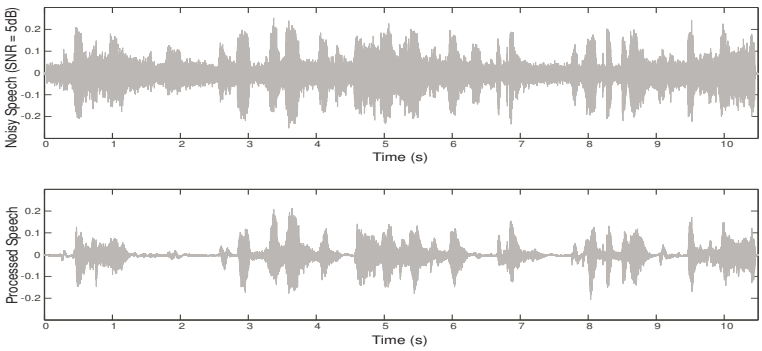


(c)

Figure 11: **Squaring Non-linearity:** (a) Schematic of squaring block implemented with MITE elements in the FPAA. (b) Transfer function of single channel system implementing the squaring non-linearity, resulting in an expansion of the dynamic range. (c) Transient response of system for a single tone input(gray trace). Large input amplitudes are amplified while smaller inputs are attenuated.

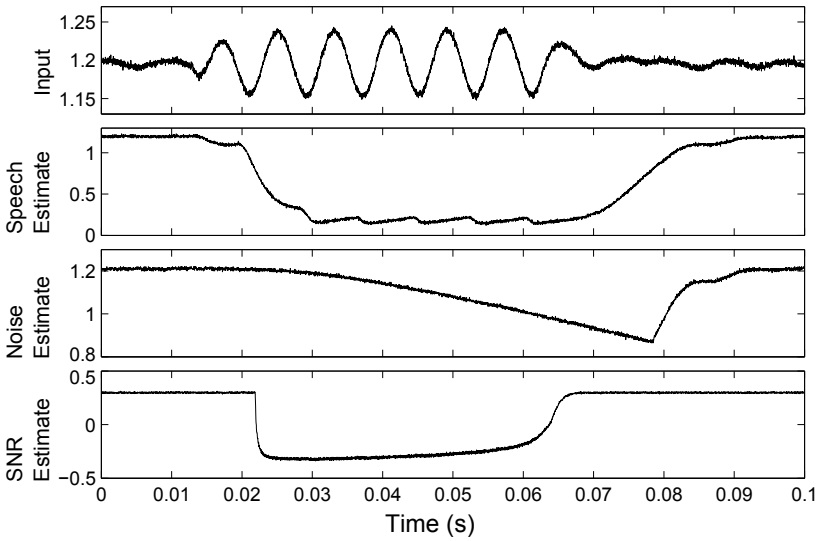


(a)

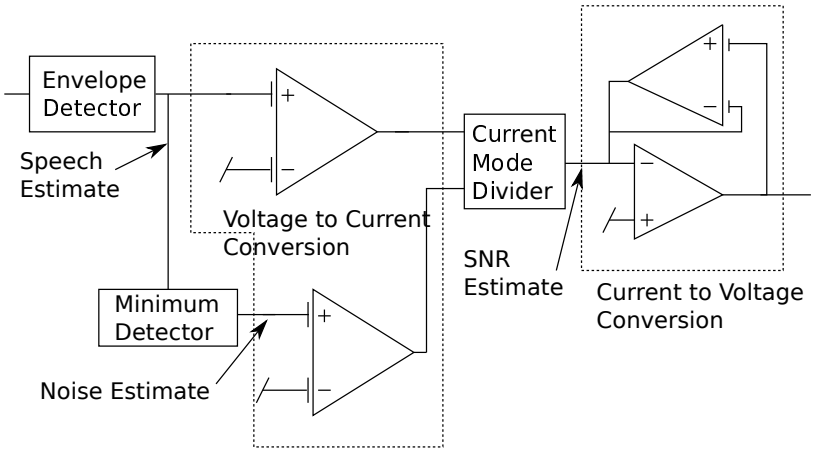


(b)

Figure 12: **Multi-channel System Results:** Comparison of noisy and processed speech for 5dB input SNR in the (a) Envelope thresholding system, (b) SNR estimation system.

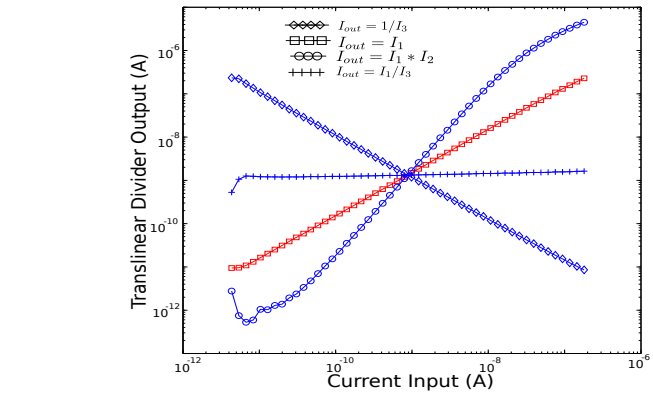


(a)

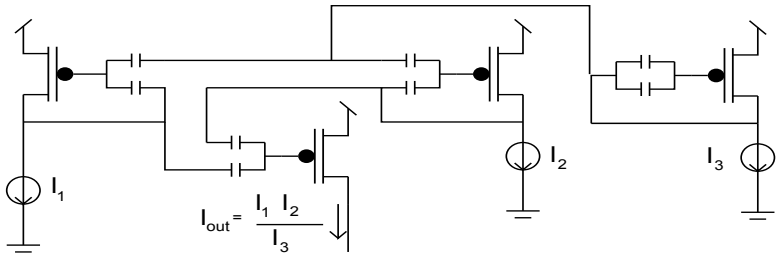


(b)

Figure 13: **SNR Estimation System:** (a) The input and control voltages generated in the SNR estimation system. (b) System block diagram: The SNR estimate is computed by using the speech estimate and the noise estimate. The speech estimate is approximately equal to the envelope while the noise estimate is equal to the slow average of the minimum of the envelope. The current mode divider requires the conversion of the estimates from voltage to current domain. This is achieved using FGOTA blocks. The divider output is converted back into voltage using a transimpedance amplifier.



(a)



(b)

Figure 14: **SNR Estimation System:** (a) Characterization of the trans-linear multiplier/divider circuit with the relation $I_{out} = I_1 * I_2/I_3$. The slope when I_1 and I_2 are swept together is double that of when I_1 is swept alone. When I_3 is swept with I_1 and I_2 kept constant, the slope is -1. (b) Implementation of the current mode multiplier/divider circuit on the FPAA using MITE transistor elements.

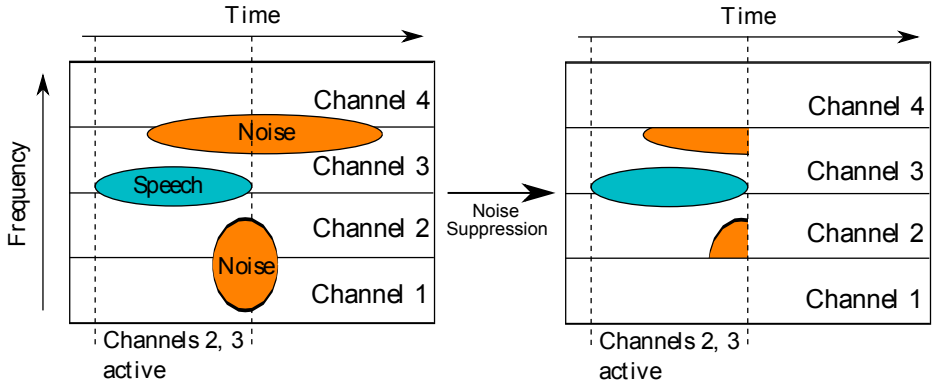
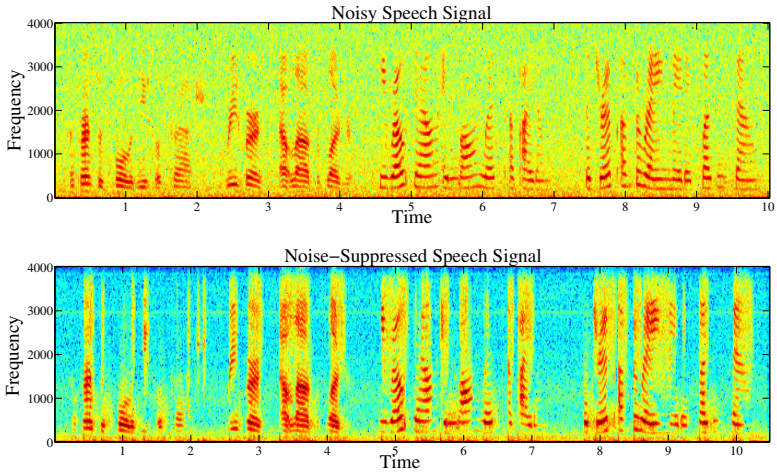
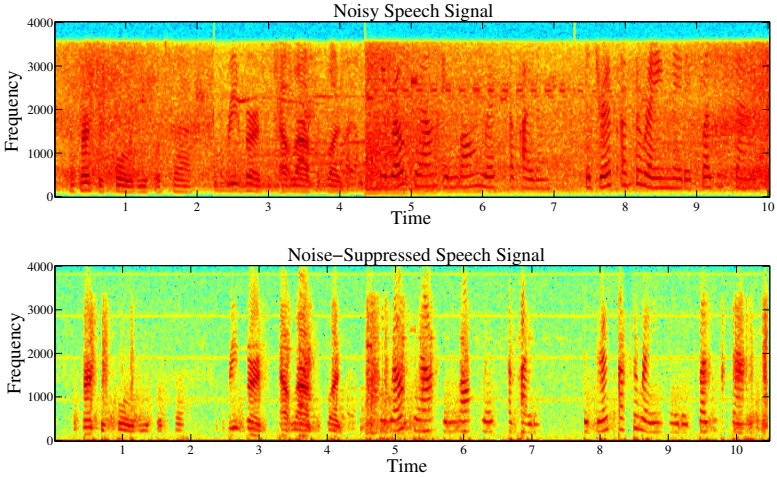


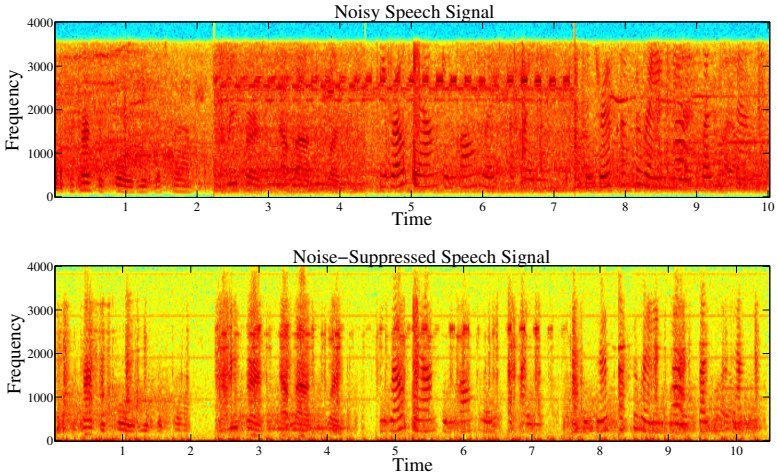
Figure 15: **Effect of limited channels on performance:** The gain control system is active when speech activity is detected, transmitting the entire channel for that duration. Any noise present in the active channel in that duration is also transmitted to the output.



(a)



(b)



(c)

Figure 16: **Spectrogram of noisy and processed speech:** Comparison of noisy and processed speech for (a) Speech with added pink noise, at 13 dB SNR, (b) with street noise at 5dB SNR, and (c) with added pink noise, at 0 dB SNR

CHAPTER III

THE VMM AND WTA AS AN ANALOG CLASSIFIER

In embedded systems that receive sensory inputs, process and classify them to make decisions, it is essential to take a low-power approach for enabling such structures in robots and other mobile platforms. Classifiers are typically used in the information refinement stage and it is often essential that besides being low power, they also produce very few events. Events are generated when a certain class has been detected, triggering further circuitry dependent on this decision. Energy efficiency is a key concern in information processing in low-power smart sensors and mobile devices [69]. A typical information processing chain usually involves a refinement stage that reduces the processing load on the following stages.

In highly integrated systems, an increased number of events often leads to increased power consumption, which is required to transmit events over interconnects between blocks that have significant capacitances. This strategy of minimizing the number of events is observed in biology, where the nervous system processes several sensory inputs and refines the information before transmitting them along large distances. The high power efficiency of the nervous system observed in biological organisms, is achieved by maintaining a low rate of spiking in the neurons, which is on average 100 Hz or less [71].

There are advantages to using Analog Signal Processing (ASP), as opposed to digital, for classification tasks that do not require high precision [94]. In the past, significant effort in hardware classifiers has been through the rise of the Artificial Neural Network (ANN) community since the 1980s, which solidified a framework of neural models that resulted in a variety of techniques to solve problems in many

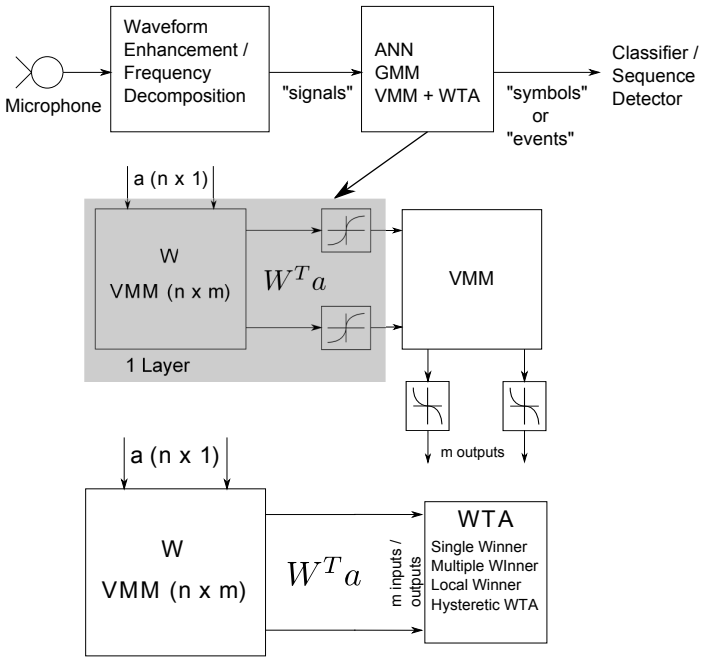


Figure 17: **Application in Analog Speech Recognizer Chain:** The speech input undergoes frequency decomposition or enhancement resulting in sub-band signals. These signals undergo first-level information refinement in the feature detection stage, resulting in a sparse "symbols" or "event" representation. The following stage detects sequences of symbols/events to identify words or syllables. The feature detect stage maybe implemented as an Artificial Neural network, Gaussian Mixture model (GMM) or a VMM+WTA classifier. A typical 2-layer NN has synaptic inputs represented by the VMM and the sigmoid modeling the soma of a point-neuron. Alternatively, we can have synaptic computation followed by a competitive network modeled by the WTA. We investigate computational advantages to using the VMM+WTA over the ANN/GMM approach.

applications. Many of these techniques are considered standard and taught in most universities. The NN approach has its early roots in the perceptron [91] and adaptive filter models [110] that then extend to multi-level network models, Hopfield models as well as other related computational approaches.

In the simplest approach, we have inputs being multiplied by a weight vector, added together at the soma compartment, where a linear or nonlinear function is applied before we receive the output. ANN approaches include having continuous valued (e.g. $tanh$) functions that approximate the spike frequency versus current

input (f-I) characteristic of neurons with an analog voltage, or spiking (integrate-and-fire neurons, rate-encoded neurons), feedforward or feedback stages.

In this research, we consider an analog classifier consisting of a Vector-Matrix Multiply (VMM) terminated with a Winner-Take-All (WTA), shown in Fig. 17, that is versatile and has more computing power than a 1-layer NN. The VMM block performs a multiply operation between a vector and a matrix of weights, resulting in a vector and forms a core component of many signal processing algorithms. The VMM+WTA, which we use as the base classifier, compares favorably against the 1-layer NN in terms of the number of components as well. We show a direct translation of a 1-layer NN to a VMM+WTA, where the WTA acts as a current comparator. In a different formulation, the WTA can perform an analog *max* function, selecting the largest (or smallest) of its inputs. With minor modifications, the WTA can be designed to allow multiple winners, local winners or exhibit hysteresis [46, 56, 75], leading to classifiers that allow multiple winners with spatial responses which can be useful in image processing, or exhibit hysteresis which makes the classifier immune to noisy inputs.

We see this structure being used in an analog speech recognizer as shown in Fig. 17. The speech input undergoes frequency-decomposition or signal-enhancement in the front-end, resulting in input features such as sub-band energies. These signal inputs are transformed into symbols or events with ANN, GMM or VMM+WTA in the first stage of information refinement. This can be followed by higher level refinement or by a sequencing block to detect syllables or words.

This chapter is organized as follows. We briefly discuss the computational efficiency and circuit complexity comparisons of VMM+WTA versus NN implementations in Sec. 3.1. In Sec. 3.2, we describe the hardware platform used for implementing our classifiers. Next, in Sec. 3.3, we discuss the WTA circuit, its dynamics, and modifications made to obtain a multiple-winner WTA. In Sec. 3.4, we describe

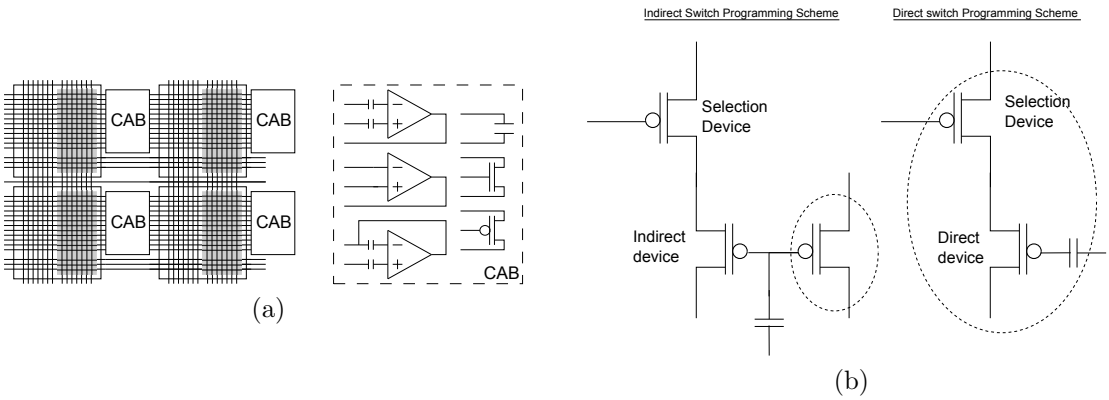


Figure 18: **Field Programmable Analog Array (FPAA):** The FPAA used in this work consists of 78 Computational Analog Blocks (CABs) embedded in reprogrammable routing enabled by floating-gate switches [98]. Each CAB consists of capacitors, transistors and Operational Transconductance Amplifiers (OTAs) that have programmable bias currents. Some OTAs have floating-gate inputs that allow cancellation of input offsets. The routing elements may be of the two types shown in (b). **Switch programming schemes:** The device(s) within the dashed circle appear in the signal path while other devices are used for programming and selection. The indirect programming scheme minimizes parasitic in the signal path by using a separate device that shares the floating gate with the actual device. The selection device is required for isolation. The indirect scheme can result in inaccuracies due to mismatch between programmed device and actual device, but can be characterized. The direct scheme, where the programmed device and actual device are the same, requires no additional characterization. However, there is an extra selection device in the signal path which reduces switch conductance at low voltages.

our VMM implementation, which is more compact and has lower noise and power than the previously described VMMs. In Sec. 3.5, we present measured results from classifier circuits that integrate the VMM and WTA to yield linear, multi-class and nonlinear classifier systems. Finally, we discuss mismatch, computing efficiency, and temperature effects in Sec. 3.6.

3.1 Implementation and Efficiency Overview

A 1-layer neural network requires the computation of a Vector-Matrix Multiply (VMM) + neuron. The addition of various weighted inputs is achieved through Kirchoff's Current Law (KCL) at the soma node, adding all currents. We define synaptic computation as the multiplication of inputs with synaptic weights, and neuron computation

as a non-linear threshold function. Assuming we have n synapses per neuron and m neurons, we expect a complexity of $m * n$ for synaptic computation. The computation at the neuron is governed by the choice of complexity in the model. A simple neuron model ($tanh(.)$) would require 4 MAC (Multiply ACcumulate) per neuron computation, as seen from a Taylor series expansion with 4 terms.

$$\begin{aligned} tanh(x) &\approx x - \frac{x^3}{x} + \frac{x^5}{5} - \frac{x^7}{7} \\ &\approx x(1 - \frac{x^2}{3}(1 - \frac{3}{5}x^2(1 - \frac{5}{7}x^2))) \end{aligned} \tag{9}$$

Usually, for moderate size of n, the synaptic computation dominates the neuron computation.

The VMM+WTA classifier topology has the advantage of being highly dense and low power. Each multiply is performed by one single transistor that stores the weight as well, and each WTA unit has only 2 transistors, providing very high circuit density. Custom analog VMMs have been shown to be 1000X more power efficient than commercial digital implementations [97]. The non-volatile weights for the multiplier can be programmed allowing flexibility. The transistors performing multiplication are biased in deep sub-threshold regime of operation, resulting in high computing efficiency. We combine these advantages of VMMs with the reconfigurability offered by FPAA platforms to develop simple classifier structures. VMMs on FPAA with high power efficiency have already been demonstrated in core signal processing applications viz. Image transforms and OFDM receivers [17, 106]. In this chapter, we discuss the computing power of the VMM+WTA classifier, and show that we can implement any 2-layer perceptron with modifications to the WTA.

3.2 Hardware : FPAA Implementation

The hardware platform used for implementing the classifier is among the family of Field Programmable Analog Array (FPAA) chips, specifically geared towards building large VMMs. A detailed description of this chip and its features can be found in

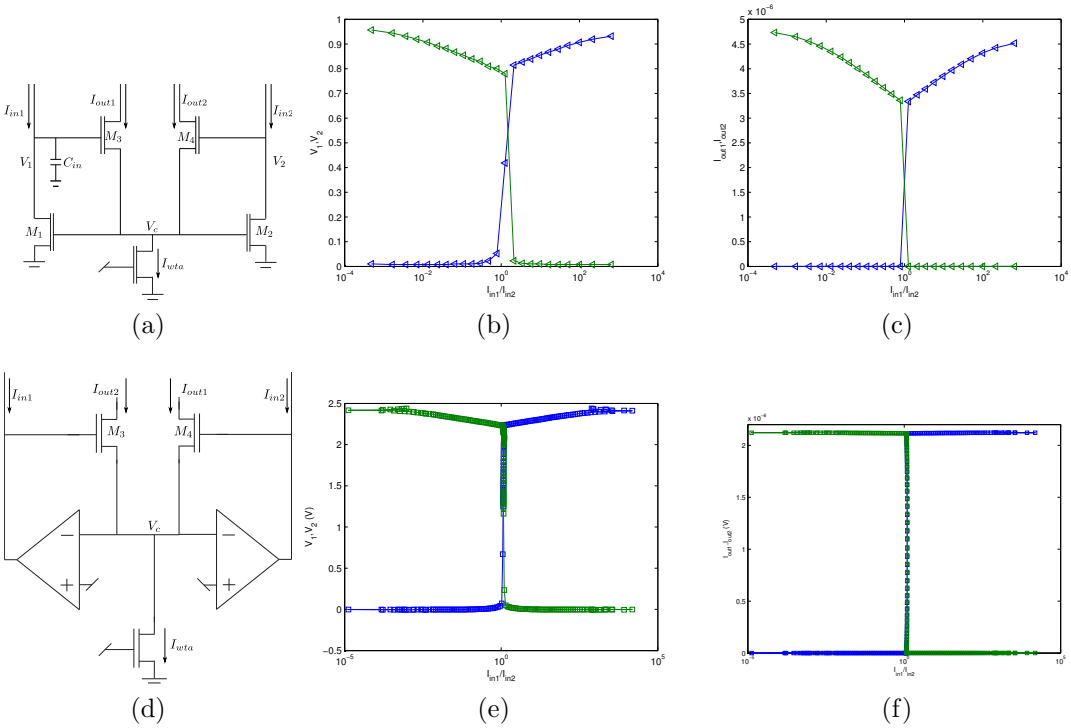


Figure 19: **Schematic of winner-take-all structures and its input-output characteristics:** (a) Circuit diagram of current-mode Winner-take-all structure from [56]. (b) Voltage outputs on the winner-take-all input nodes for a differential input current. (c) Current output of the winner-take-all for a differential input current. (d) Circuit diagram of the modified winner-take-all using OTAs. (e) Voltage outputs on the winner-take-all input nodes. (f) Current outputs from the winner-take-all.

[98]. However, for the sake of completeness, we provide a short discussion on the architecture of this chip.

FPAAs have the general structure of Computational Analog Blocks (CAB) with routing infrastructure to make re-programmable connections between the components. The CAB consists of circuit blocks commonly used in analog design, as shown in Fig. 18. The re-programmability is enabled using floating-gate transistors that can be programmed ON or OFF by operations known as injection and tunneling respectively, similar to programming EEPROMs. The programming infrastructure that includes selecting specific switches and injecting them is integrated on-chip, as discussed in [6].

The chip that is used in this work, follows a similar architecture to [6] with some key differences, which are leveraged for building classifier structures. To build large VMMs, we require a large number of floating-gate switches that can be programmed accurately. The switch used for programmable connectivity uses the indirect programming structure, shown in Fig. 18b, that was developed to minimize parasitic resistance on the switch by using a separate device that shared the gate. This other device required extra devices for ensuring isolation during injection. In building VMMs, we require precise control over the weights programmed on the transistors, but this structure also suffers from mismatch issues that involve lengthy characterization process. In the FPAA used in this work, a portion of the routing switches are directly programmed switches, where the actual device used in the circuit is programmed. This removes any errors due to mismatch, but these switches are poorer than the indirectly programmed switch, since they contain an extra device in the signal path that is needed during program time for isolation purposes. The two programming schemes are shown in Fig. 18b. A detailed discussion on direct and indirectly programmed switches can be found in [30]. The components in the FPAA that are available for building classifiers are shown in Fig. 18. A high-level system overview of the VMM+WTA circuit is shown in Fig. 20. The inputs to the classifier are voltages, while the outputs from the VMM are uni-directional currents. The WTA may produce voltage or current outputs.

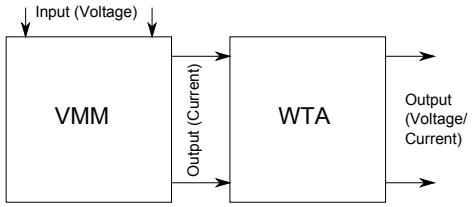


Figure 20: **High level System Flow:** The VMM takes voltage inputs and has uni-directional current output. The WTA may generate voltage or current outputs.

3.3 Winner Take All

WTA networks of neurons was an early area where VLSI and neuroscience positively interacted with each other, providing a unique and efficient means of computation. The WTA module is used for modeling competition in neural networks, specifically in representing the mechanism of attention [51]. A WTA network consists of multiple (m) somas that all connect (through excitatory synaptic connections) onto a single neuron that provides inhibitory feedback to all the original somas. The net effect is that we have an adaptive threshold, which can be global or local, that is the largest of some function on the inputs. Whether these somas are continuous valued or spiking representations is dependent on the design and computing environment.

In spiking representations, the WTA models the inhibiting effect exerted by neurons over the firing rate of competing neurons in a network. The neuron that starts to fire earliest, inhibits its neighbors before they can fire through inhibitory synaptic connections. This mechanism is critical to reducing firing rates in biological networks, which in turn translates to a fewer number of events and therefore, lower overall power dissipation. The classic circuit implementation by Lazzaro et al [56] was based on

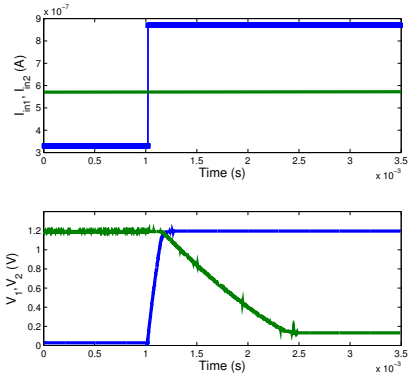
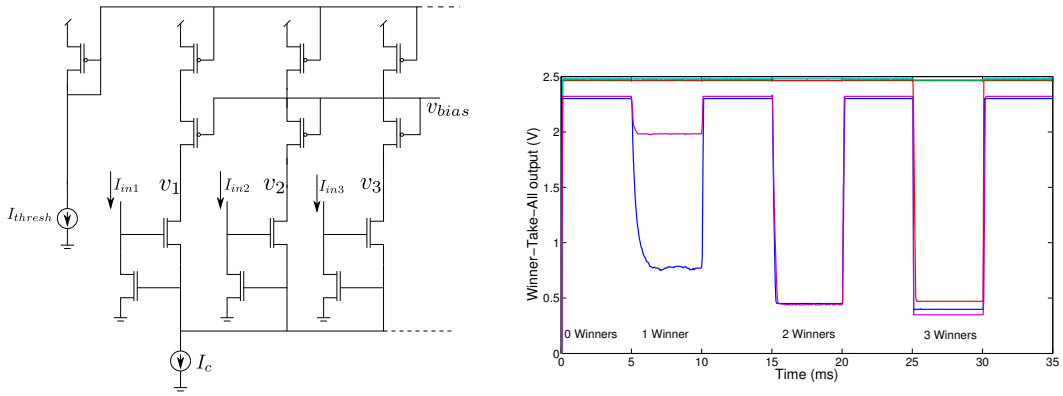


Figure 21: **Measured dynamic response of WTA to input current step:** The winning node settles faster than losers. The dynamics of the winning node are governed by a diode time constant (small). In this measurement, the WTA nodes see a large pin capacitance of $\approx$ 10pF.



(a) (b)

Figure 22: **k-winner-take-all:**(a) The traditional WTA can be modified to a k-WTA with a current threshold at each output, realized using a cascoded pFET. The current flowing through the winning branch is constrained, allowing other inputs to the WTA to win. The voltage outputs from the WTA are inverted and a node wins when its output is below mid-rail. (b) Choice of current threshold determines the number of winners: winner-take-all with $0, 1, 2$ and 3 winners.

continuous valued elements, that utilized transistor device physics to build an efficient circuit. Later, others built multiple spike-based representations to complete the connection between the analog VLSI approach and biological computation [45]. Several modifications to this circuit exist, that allow local winners, hysteresis behavior that stabilize the outputs, temporary winners that fatigue after a period of winning and allow other inputs to win and multiple winners [46, 55, 109]. In this chapter, we discuss the computational advantages offered by WTAs in classifier architectures.

The CAB components in the FPAA support several WTA implementations, but we first implement the classic WTA composed of discrete transistors shown in Fig. 19a. The two possible outputs from the WTA circuit are the voltages at the input nodes (V_1, V_2), or output currents in each branch (I_{out1}, I_{out2}). The measured DC characteristics are plotted in Fig. 19b and Fig. 19c. The WTA biases itself depending on the input currents. The small-signal gain for the WTA shown in Fig. 19a is $\frac{U_T}{\kappa I}$ where I is the DC input current, U_T is the thermal voltage and κ is the inverse of the sub-threshold slope.

Next, we implement an OTA-based WTA circuit shown in Fig. 19d respectively. The transistors and OTAs are CAB components in our chip, effectively resulting in the same number of components. However, there are trade-offs in using either implementation. The OTA-based WTA can take bi-directional input currents, while the classic WTA takes uni-directional current inputs. The OTA-based WTA has the constraint that the tail current of the OTAs have to be set larger than the largest expected input current into the WTA. The OTA-based WTA circuit has a gain dependent on the OTA bias current given by $\frac{2U_T}{\kappa I_{bias}}$, where I_{bias} is the tail current of the OTA. Increasing I_{bias} results in an increased dynamic range of inputs to the WTA, but also reduces the gain of the WTA.

3.3.1 WTA dynamics

The winner-take-all performs a highly non-linear computation which results in different settling behaviors for the winning and losing nodes. The initial state of the winner-take-all and the magnitude of the differential inputs determine the time constant of the settling nodes. In the case of the two-input circuit shown in Fig. 19a, if $I_{in1} > I_{in2}$, M_1 is in saturation and behaves like a diode-connected nFET while M_2 is in the ohmic region of operation. Further increases in I_{in1} result in quick settling of V_1 since it is a low-impedance node with a time constant of $CU_T/(\kappa I_{in1})$. Now, if I_{in2} changes to become greater than I_{in1}, once V_2 charges up to move M_2 out of ohmic and into saturation, it charges up at a rate determined by the Early voltage V_A. As an input starts to lose, for small difference between inputs, the input node of the winner-take-all undergoes a high impedance $\frac{V_A}{I_{in}}$ phase, resulting in a slow transition. For large difference between inputs, the input becomes a low-impedance node with the transistor in linear region with an impedance of $\frac{U_T}{I_{in}}$. As an input starts to win, the input node undergoes a transition from low impedance to high impedance back to low impedance. In this circuit, the winners settle faster than the losers, as seen in

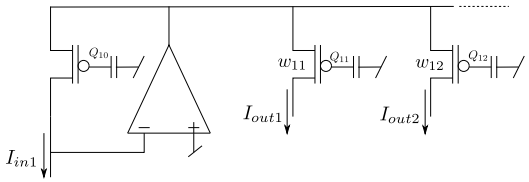
Fig. 21. This is expected from the time constants of the winning and losing nodes. The time response is slower when the inputs are close to each other and then change. When the inputs are far apart and then change, the response is faster.

3.3.2 Multiple Winners

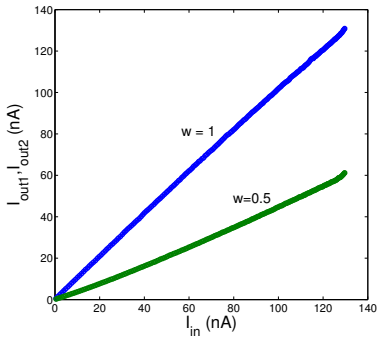
Often, we require classifiers that generate not just one output, but multiple outputs. In pattern classification, we can expect the classifier to indicate that a certain pattern matches two categories instead of just one. The WTA circuit does not preclude multiple winners and this can be achieved by modifying the circuit shown in Fig. 19a. For a k-WTA, or a WTA with k winners, we use the current outputs from the WTA and apply a current threshold at the output. The modified implementation is shown in Fig. 22a. A current threshold I_{thresh} is mirrored to each of the current outptus from the WTA. By constraining the current in the winning branch, we allow other inputs of the WTA to continue winning after the first winner. The choice of I_{thresh} determines the number of winners. For k winners, the relation between I_{thresh} and I_c is given by

$$\frac{I_c}{k+1} \leq I_{thresh} < \frac{I_c}{k} \tag{10}$$

The distribution of input currents also determines the number of winners for a fixed I_{thresh}. When the inputs are close to each other, I_{thresh} needs to be closer to $I_c/(k+1)$ than I_c/k. The value of I_{thresh} required to guarantee k winners is given by the lower limit of (10). Fig. 22b shows the measured results from a five-input WTA, with different current thresholds to obtain multiple winners. The cascoding pFET devices were inserted to improve the Early voltage of the current threshold, thereby constraining the current through the winning branches to I_{thresh} more effectively than if cascodes were not present. The k-WTA produces inverted voltage outputs that are taken at the drain of the thresholding pFET. Compared to the k-WTA circuit in [109], this implementation does not require any additional power/circuitry.



(a)



(b)

Figure 23: 1**x**2 **VMM characterization:**(a) Schematic of a 1x2 VMM with current inputs, as described in [97]. The OTA with base floating-gate is a logarithmic trans-impedance amplifier and generates a source voltage that is applied to other devices with programmed weights. (b)measured results from a 1x2 VMM programmed to weights of 0.5 and 1.

3.4 Compact VMM Implementations

VMMs can be implemented in a power-efficient and compact manner using floating gates. The multiplication weights are stored as charge on the floating node and can be precisely programmed and controlled. The weight can be expressed as

$$w = e^{\kappa Q / C_T U_T} \tag{11}$$

where Q is the charge programmed on the floating-gate node and C_T is the total effective capacitance seen at the floating node. A single floating gate stores the weight as well as performs a multiply function. The programming accuracy of the VMM weights have been well characterized and in one application, has been shown to be 1.5% accurate in [98]. Examples of the different VMM topologies that we can implement are discussed in [97]. The schematic of a 1x2 VMM which achieves single-quadrant multiplication is shown in Fig. 23a. This circuit takes uni-directional

current inputs and has positive weights. The core of the multiplication is achieved using a current mirror structure. However, in contrast to a traditional current mirror where the gate terminal is broadcast, we use a source broadcasting topology that is more conducive to implementation on our reconfigurable chips. The weights w_{11} and w_{12} are programmed by setting a difference in charge $Q_{11} - Q_{10}$ and $Q_{12} - Q_{10}$, and can be expressed as

$$w_{11} = e^{\kappa(Q_{11}-Q_{10})/U_T} \tag{12}$$

$$w_{12} = e^{\kappa(Q_{12}-Q_{10})/U_T} \tag{13}$$

Fig. 23b shows the measured current outputs from the VMM programmed with weights 0.5 and 1. To achieve four-quadrant multiplication, we require a VMM that takes differential inputs and implements signed weights. These structures are discussed in [97].

For a VMM with voltage inputs, we require a structure shown in Fig. 24a. This structure converts a voltage input linearly into a current, using an OTA as a trans-conductance stage. The current is log-compressed on the source terminal using a logarithmic trans-impedance amplifier and broadcast. Hence, the voltage input into the VMM and the weight of the input is encoded in the source voltage using the trans-conductance amplifier and logarithmic trans-impedance amplifier. This signal-conditioning block, that maps input voltage to a broadcasted source voltage is shown in the dashed box in Fig. 24a.

We note that the number of OTAs required in the signal-conditioning block scales linearly with inputs, and is $2n$ for single-quadrant multipliers, and $4n$ for four-quadrant multipliers, where n is the number of inputs. Hence, for classifiers with a large number of inputs, a significant portion of the power budget is spent in the signal-conditioning block. In addition to the power overhead due to the amplifiers, we see effects of mismatch and noise added on the inputs. The main sources of mismatch

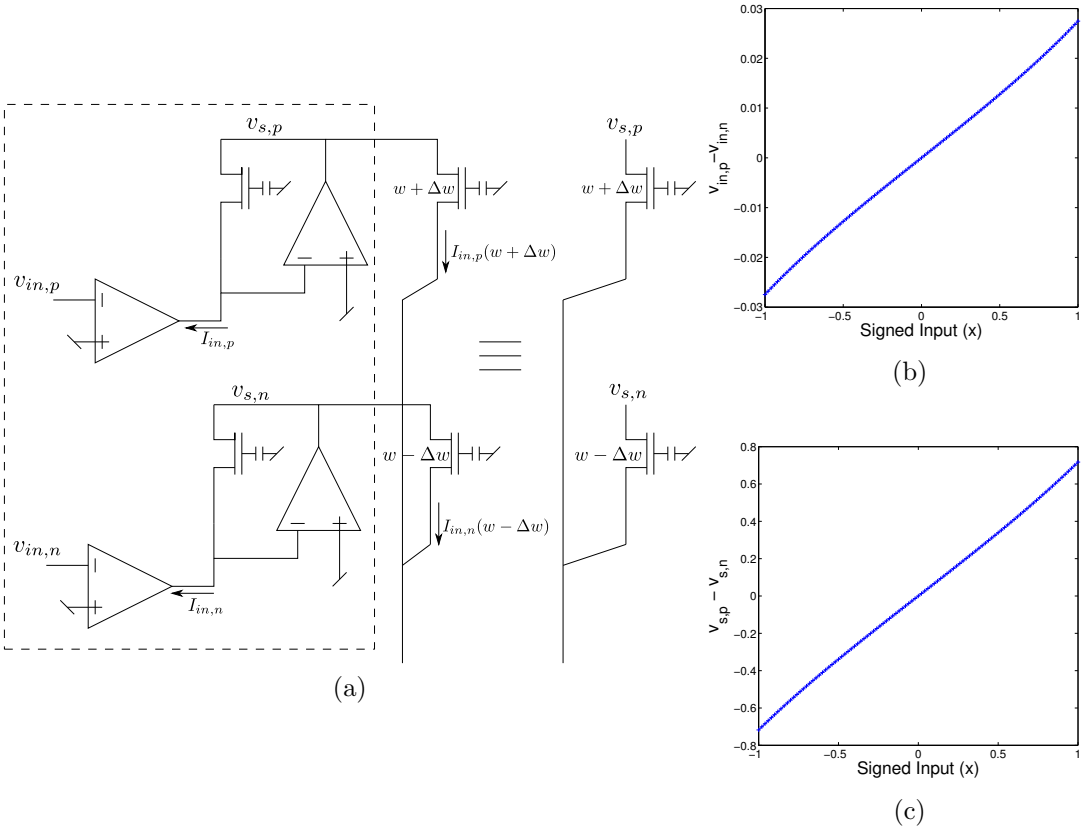


Figure 24: **Equivalence between VMM topologies:** We show that the two VMM circuits are equivalent in the Appendix. The VMM structure shown on the left transforms voltage inputs into currents using a trans-impedance stage. The multiplication is achieved in current mode using a source-driven floating-gate current mirror where the weights are a result of difference in charge programmed on the floating gates. $v_{in,p} - v_{in,n}$ is the differential input to the VMM. The structure shown on the right shows a source-driven VMM where the input signal is applied directly to the source of the floating gates. $v_{s,p} - v_{s,n}$ is the differential input to the source-driven VMM. (b) and (c) plot the differential input voltages for the two topologies versus signed input x, as calculated in (66). x is the normalized input to the classifier. We notice that the input for the source-driven topology is a compressed form of the input for the standard VMM topology. In this simulation, we use $\kappa_{eff} = 0.05$.

are input offsets in the V-I, mismatch between bias currents of the V-I, and input offsets in the I-V. Cancellation of these effects often requires a lengthy characterization process.

In this work, we choose a source-driven VMM topology as shown in Fig. 24a to build low-power compact structures that minimize the added noise and mismatch

effects by eliminating 4 OTAs per VMM input. Here, the voltage inputs are directly applied the source terminal of the weighted current sources. We derive the equivalence between the two topologies in the Appendix.

We assume that the inputs to the classifier are from the set $\{x : |x| \leq 1\}$, which is reasonable for normalized inputs. It can be shown that for small x, there is a linear relation between the differential inputs for the two different VMM topologies shown in Fig. 24a.

$$v_{in,p} - v_{in,n} \propto v_{s,p} - v_{s,n} \tag{14}$$

The differential input at the source is a compressed linear representation of the inputs to the V-I, and the attenuation factor is inversely related to the input linear range of the trans-conductance stage. The two voltage inputs for different values of x is plotted in Fig. 24b.

The equivalence of the two structures in Fig. 24a shows that we can achieve compact VMM structures using just the routing infrastructure in the FPAA. From (14), the voltage inputs to the VMM can be applied directly to the source of the FG transistors. We note that the differential voltage inputs to the source-driven VMM need to be constrained to a range of $2U_T \approx 50$mV for linear operation of the VMM. The stage driving the VMM also needs to supply the current required for the VMM. The output current can be expressed as

$$I_{out} = I_{bias}(2w + x\Delta w) \tag{15}$$

3.5 Capability of VMM+WTA Classifiers

We now integrate the VMM and WTA circuits to build simple classifier structures. In this section, we first describe measured results from system compilations of linear, multi-class and non-linear classification problems.

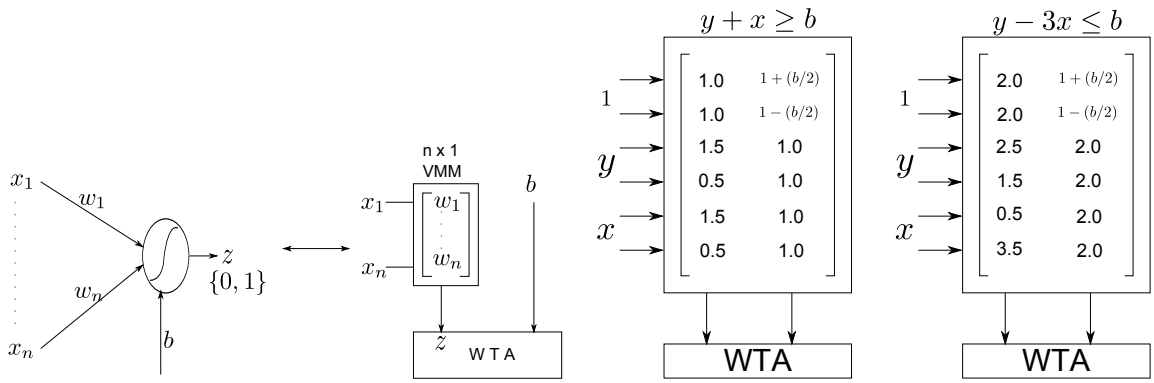
3.5.1 Linear Classifiers

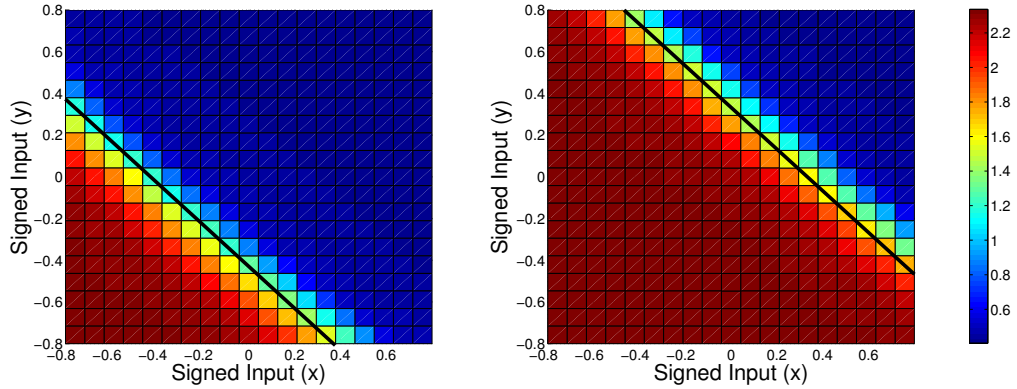
We start by considering a perceptron, which is a simple linear classifier with a binary output that can be implemented with a 1-layer neural network. A linearly separable set of inputs can be classified using a perceptron trained to weights w_i and bias b having the equation

$$z = \begin{cases} 1 & \text{if } \sum_i w_i x_i - b \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

A VMM+WTA classifier can be trained as a generalized single-layer perceptron by using a fixed current source as an additional bias input to the WTA, shown in Fig. 25a. The WTA functions as a current comparator and the detects the larger of the inputs. When $\sum_i w_i x_i > b$, the first input wins. By using a 1-WTA circuit implemented with the current threshold at the WTA output, we obtain inverted voltage outputs. Hence, the first output is low when $\sum_i w_i x_i > b$ and high otherwise.

We measured results from two different linear classifier boundaries programmed on the VMM+WTA circuit, for multiple bias values. For a linear decision boundary, we train a perceptron using MATLAB's Neural Network Toolbox and apply the weight and bias values directly to the VMM+WTA classifier. We restricted ourselves to a 2-input case for ease of visualization. The structure in Fig. 25a only supports positive values for the bias. Since our implementation required signed weights and bias values, we chose a topology with fully-differential inputs. The classifier was tested over all inputs from the set $\{(x, y) : |x| \leq 0.8, |y| \leq 0.8\}$. We plot the inverted WTA voltage output in Figs. 25c, 25d. The output makes a sharp transition at the desired decision boundary, which is marked by the solid line in the plots. Since our VMM implementation consisted of directly programmed floating-gate transistors, we were able to directly apply the weights obtained from the training algorithm and target them to the hardware, without any calibration or offset correction procedure and still match the theoretical decision boundary.





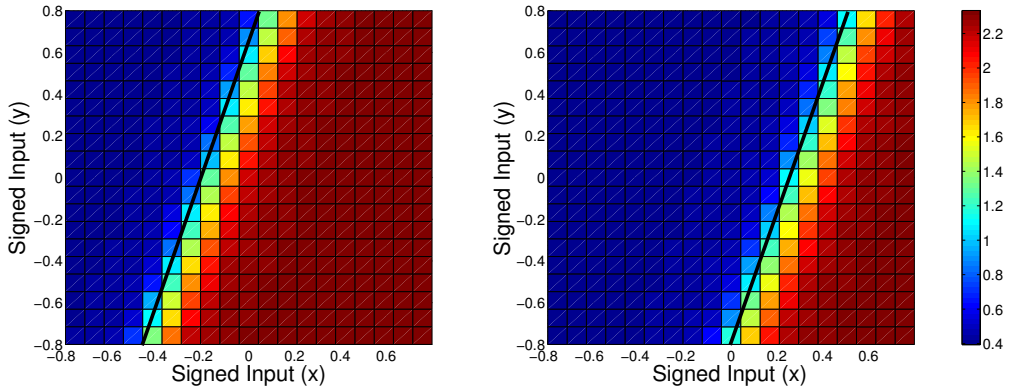
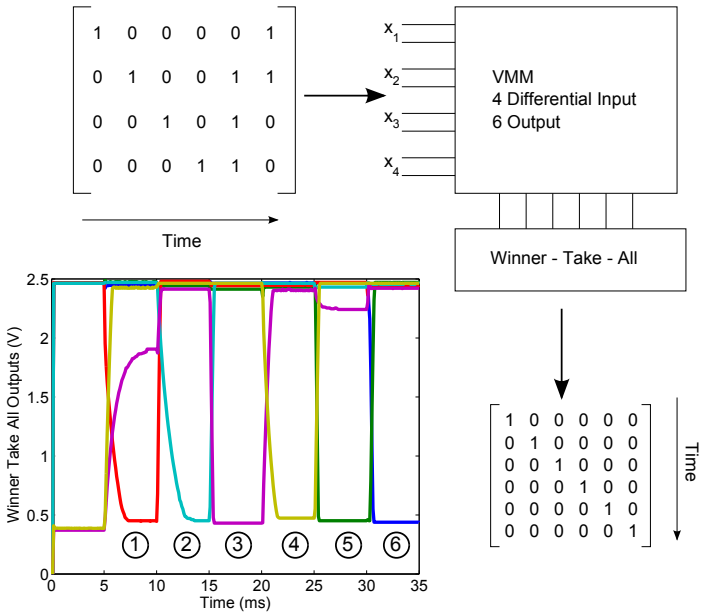
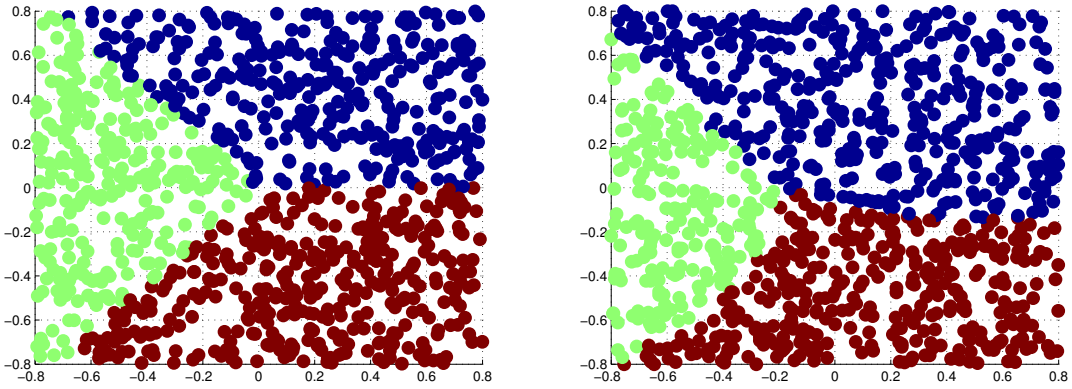


Figure 25: **Linear Classifiers:** A simple perceptron or a one-layer feed-forward network can be implemented using a VMM+WTA structure. (a) The input multiplication can be implemented using VMMs. The bias b is the second input to the WTA, implemented as a fixed current source. (b) Differential implementation of linear separator. The bias is programmed as a differential weight with a fixed input. **Measured results:**(c) A VMM+WTA classifier trained to have a decision boundary of $y + x \geq b$, for bias values $b = 0.25, -0.25$. (d) VMM+WTA classifier trained to have a decision boundary of $y - 3x \leq b$, for bias values $b = 0.75, -0.75$. The black solid line represents the theoretical decision boundary.



(a)



(b) (c)

Figure 26: **Multi-dimensional classifiers:** (a) A four-input six-output VMM+WTA classifier constructed to classify input sequences. The weights are computed using the psuedo-inverse method. The trained classifier responds to sequence of input patterns. (b) A two-input three-output VMM+WTA classifier constructed to have the theoretical decision boundaries shown. Each color represents a different winner. (c) Measured results from the VMM+WTA classifier compiled.

3.5.2 Multi-class Classifiers

As the name suggests, multi-class classifiers have several outputs, and classify data into multiple classes. The competitive behavior modeled in the VMM+WTA circuit allows building of such classifiers with multiple outputs that can detect regions of

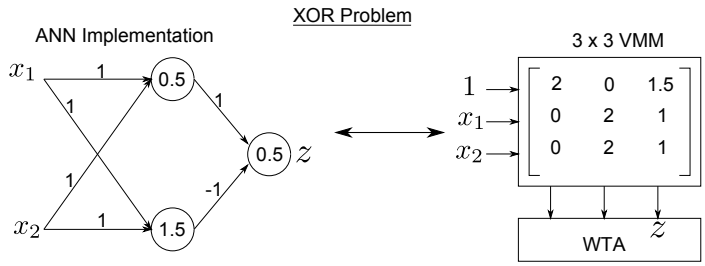
interest. We demonstrate the capability of the VMM+WTA circuit to build a region detector in Fig. 26b. We train a 2-input, 3-output classifier to detect regions of inputs defined as shown in Fig. 26b. Again, for simplicity of visualization, we chose only 2 differential inputs. We constructed a classifier with 3 outputs and the region boundaries specified in Fig. 26b. From this theoretical construction, we obtained the weights for the VMM using the pseudo-inverse method. We generate random inputs in MATLAB and multiply them by the weight matrix obtained. We then do a *max* function on the transformed inputs to generate the theoretical classifier output in Fig. 26b. Since the theoretical weights were signed, we constructed a fully-differential implementation and targeted the weights to the VMM circuit. We then applied 1000 inputs randomly from the set $\{(x, y) : |x| \leq 0.8, |y| \leq 0.8\}$. Since the WTA voltage outputs are inverted, we found the winning output by finding WTA voltages below inverter threshold(mid-rail) and recording its position. In Fig. 26c, we denote the winning position for each of the random inputs by a different colored dot. Our 3-output classifier was programmed with weights obtained directly from MATLAB. It matches the desired classifier response quite well. Multi-class classifiers are often used as pattern recognizers. We constructed a simple pattern recognizer problem consisting of 4 inputs and 6 outputs, by artificially choosing a set of inputs and outputs to the system. The desired system response is shown in Fig. 26a, where the input sequence produces an identity matrix at the output. Each column in the identity matrix represents an output of the WTA and each bit of the 4-bit input pattern represents a differential input. We obtained the weights using the pseudo-inverse method and programmed the classifier. We tested the classifier by generating a repeating sequence of the inputs using the on-board DACs on our hardware platform. Each pattern was held for 5ms before the next pattern was presented to the classifier. The transient response measured from each output of the WTA shows that the system classifies the patterns correctly. Since the outputs from the WTA were unbuffered

and saw a pin capacitance of $\approx$ 10pF, we see a slow transition between states of the classifier. This was also the reason why we presented inputs for a long time before switching, and can be avoided by buffering the output nodes.

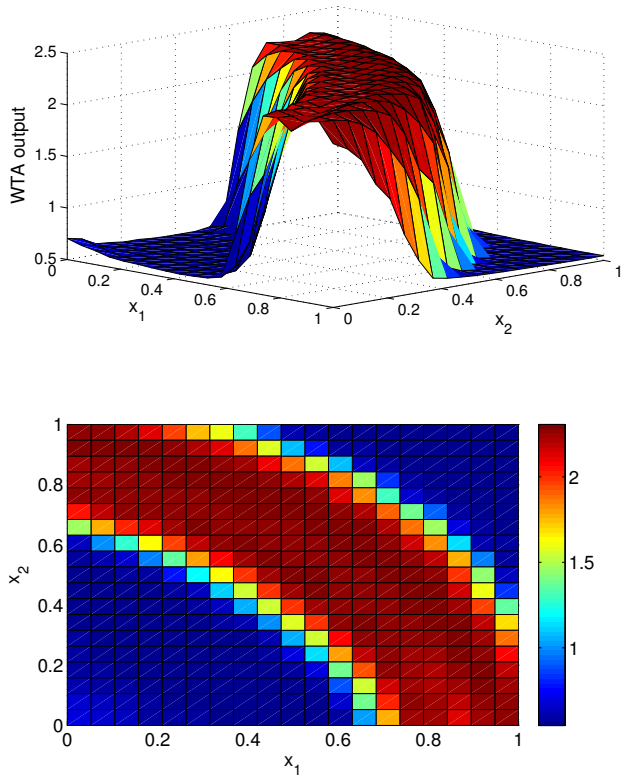
3.5.3 Non-linear classifiers

Non-linear classification boundaries required in most real-world problems are usually very computationally intensive. Single-layer neural networks can only implement classifiers for linearly separable data, but a 2-layer NN can approximate any function [101]. A 2-layer NN has an input layer, hidden layer and an output layer. An analog VLSI implementation would require 2 VMMs for the synaptic computation and 2 layers of threshold blocks for the hidden and the output layers. This considerably increases the complexity and power consumption of the circuitry. In [64], Maass showed that any boolean function with analog or digital inputs and one binary output can be approximated with a VMM+k-winner-take-all classifier. He showed that the weights for the VMM+WTA classifier are a linear combination of weights of the 2-layer perceptron, and further, they are all positive, requiring only single-ended inputs in our implementation. This result provides additional support to the computational power of the VMM+WTA classifier, by halving the computing resources required.

One of the most computationally challenging problems for neural networks is the XOR problem, which does not involve a linear decision boundary. We use the algorithm provided in [64] to compute weights for our VMM+k-winner-take-all structure to implement a non-linear classification boundary for an XOR circuit. One possible implementation of the XOR gate with a 2-layer neural network and its equivalent VMM+WTA implementation is shown in Fig. 27a. The VMM+WTA XOR circuit requires only a single-winner WTA. The position of the WTA output computing the XOR function is marked z in Fig. 27a. We tested the XOR circuit by generating inputs from the set $\{(x, y) : 0 \leq x \leq 1, 0 \leq y \leq 1\}$ and recording the voltage at the

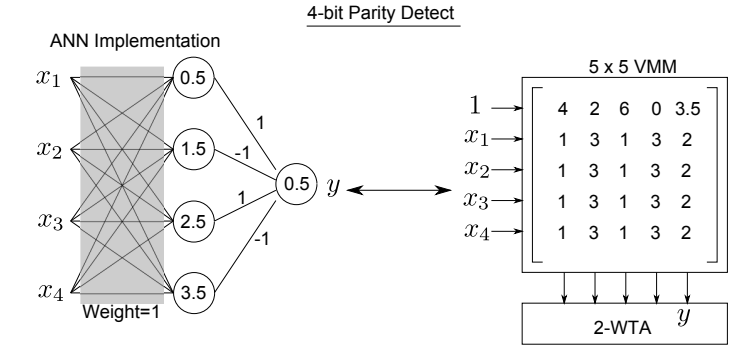


(a)

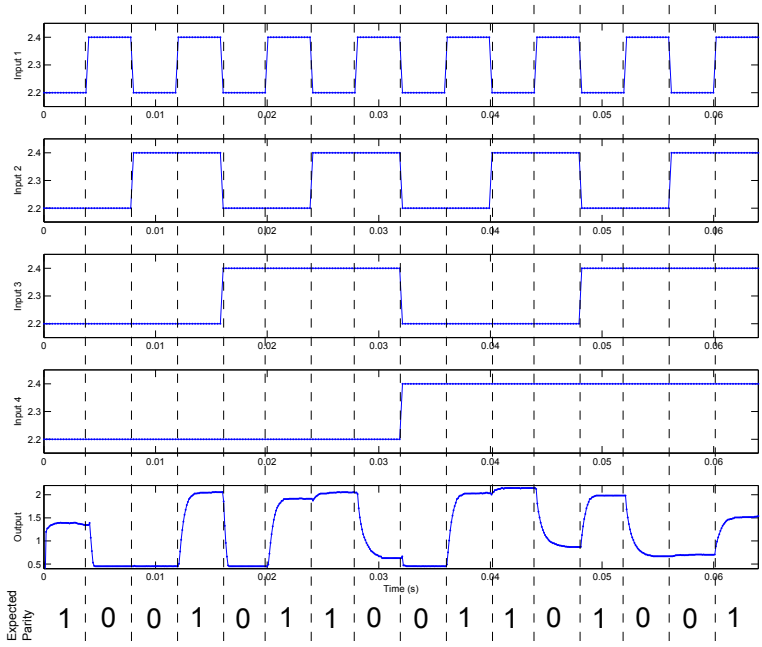


(b)

Figure 27: **Nonlinear Classifiers:** The VMM+WTA structure is powerful enough to implement any boolean function with one digital output. (a) A solution for the XOR problem using a two-layer neural network can be translated to a VMM+WTA implementation. (b) Measured results from an XOR implementation using the VMM+WTA structure.



(a)



(b)

Figure 28: **Nonlinear Classifiers:** The VMM+WTA structure is powerful enough to implement any boolean function with one digital output. (a) A four-bit parity problem implementation using a two-layer neural network and its equivalent VMM+WTA implementation. (b) Measured results from the parity detect block.

third output. The results are plotted in Fig. 27b. The VMM weights are biased at 10nA, resulting in 95nA drawn in the VMM when both inputs are active. The WTA is biased at 100nA, resulting in 0.47μW drawn at 2.4V, when all inputs are active.

The XOR gate is also the simplest case of the N-input parity function. Here, we demonstrate the implementation of a 4-input parity function using the VMM+WTA classifier. Starting with a two-layer neural network implementation [41], we obtain weights for the VMM+WTA classifier using the procedure detailed in [64]. This implementation requires a 2-WTA, with 5 inputs, with the fifth output computing the parity function. The theoretical neural network and VMM+WTA implementation is shown in Fig. 28a. We obtain a 2-WTA by setting I_{thresh} to $I_c/3$, as shown in Fig. 22, where I_c is the WTA bias current. We test the 4 bit parity circuit by setting input patterns using DACs on our test platform. We compute the expected parity (marked in the figure) and plot the fifth output from the WTA in Fig. 28b. The slow transition at the WTA output is due to the large capacitance at the node and can be avoided by buffering the output. The WTA output does not swing all the way up to the rail for the case of all zeros and all ones, since the VMM output for those cases is very close to the second winner. However, the transition from this state to the winning case is large enough, that it can be detected by a logic gate.

3.6 System performance characterization

In the following section, we characterize the system performance by considering mismatch effects, power consumption, computing efficiency and speed of computing. We also discuss the temperature dependence of the classifier output.

3.6.1 Mismatch compensation

In this section, we investigate effects of mismatch in the winner-take-all circuit, and techniques to compensate for them. We will ignore effects of mismatch in (W/L) and κ. The dominant source of mismatch in analog design is the threshold voltage

mismatch ΔV_T [52], which is true in sub- and near-threshold regions. In particular, effects of mismatch are worse in the sub-threshold mode of operation since $\delta I/I = -\kappa \Delta V_T / U_T$.

In the WTA shown in Fig. 19a, we assume that $I_{in1} = I_{in2} = I_{in}$. Then, we expect that $V_1 = V_2$. Both M_1 and M_2 share the same gate voltage V_c. The equation for the drain current through M_1, assuming sub-threshold saturation is

$$I = 2I_{th}(\frac{W}{L})e^{\frac{\kappa(V_g - V_{T0})}{U_T}} e^{V_d/V_A} \tag{17}$$

where I_{th} is the threshold current of the device, V_g is the gate voltage and V_d is the drain voltage. In the balanced case, the difference between V_1 and V_2 can be expressed as

$$V_1 - V_2 = \frac{\kappa V_A \Delta V_{T1}}{U_T} + \Delta V_{T3} \tag{18}$$

where ΔV_{T1} is the mismatch between M_1 and M_2 and ΔV_{T3} is the mismatch between M_3 and M_4. A difference in the input currents $I_{in1} = I_{in} + \Delta I_{in}$ and $I_{in2} = I_{in}$ results in a difference in output voltages given by

$$V_1 - V_2 = V_A ln(1 + \frac{\Delta I_{in}}{I_{in}}) \approx V_A \frac{\Delta I_{in}}{I_{in}} \tag{19}$$

This difference in input currents can be programmed in the VMM bias currents to cancel offsets present in the WTA. Another technique for mismatch compensation is including floating-gate transistors in the WTA circuit (M_1 and M_2), which would require floating-gate nFET devices. Our current chip does not include floating-gate nFET transistors, but this is possible in future versions of this chip. A detailed treatment of mismatch characterization and its automation on the FPAA is presented in [100].

3.6.2 Speed, Power and Efficiency

We observe the classic power-speed trade-off in the performance of the VMM+WTA classifier. The power consumption of the VMM is $\mathcal{O}(mn)$, while the WTA power

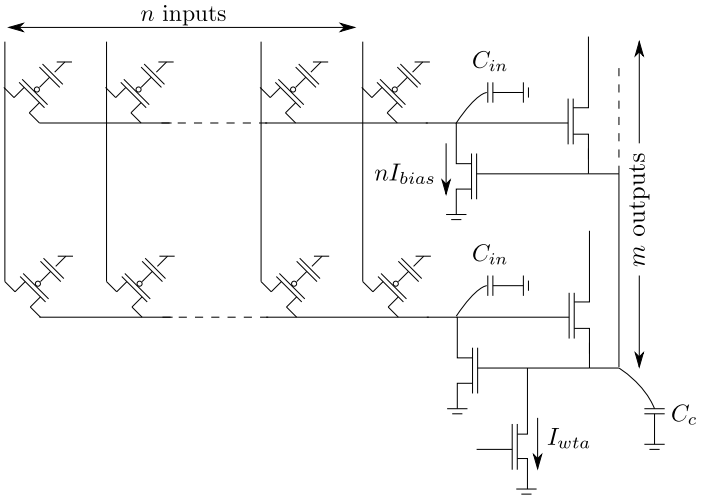


Figure 29: **Schematic of VMM+WTA circuit:** Node capacitance at the WTA input scales with the VMM inputs, the common node capacitance scales with WTA outputs. We assume single-quadrant multiplication in the VMM.

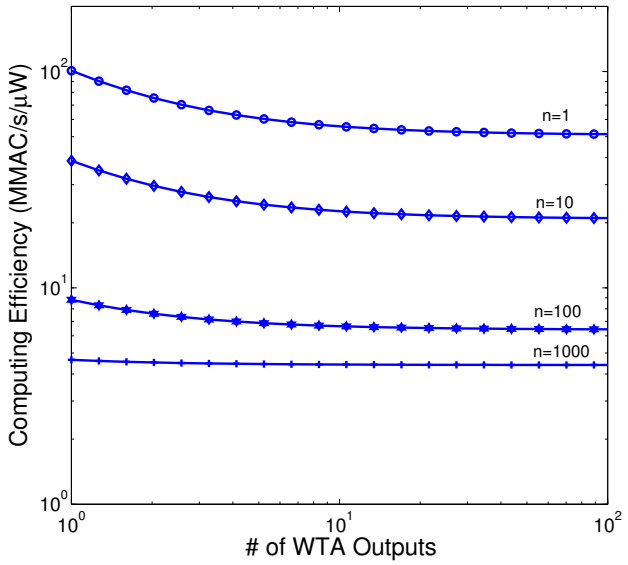


Figure 30: **Computing Efficiency vs classifier size:** The inverse of the power-delay product in (23) is approximately the computing efficiency in MMAC/s/μW, which is fixed and scales with inputs and outputs.

is $\mathcal{O}(n)$. The settling time of the WTA is dominated by the input capacitance C_{in}. The settling time can be reduced by increasing the VMM bias current, which also increases the power consumption. The dynamic response of the system is determined by the capacitance at the common node in the WTA, shown in Fig. 29. From [56],

we get first-order behavior from the circuit when

$$I_{wta} > 4nI_{bias}(C_c/C_{in}) \tag{20}$$

which gives us the WTA bias current to avoid ringing at the winning outputs. Then, the winning node has a time constant $\tau = C_{in}U_T/(nI_{bias})$. Since C_{in} scales with the number of inputs n, we write $C_{in} = nC_{in0}$. Hence, the settling time for the winning node is independent of n and can be written as

$$\tau = \frac{C_{in0}U_T}{I_{bias}} \tag{21}$$

The power consumption for our classifier, when all inputs are active can be expressed as

$$\begin{aligned} P &= P_{VMM} + P_{WTA} \\ &= mnI_{bias}V_{dd} + I_{wta}V_{dd} \\ &= I_{bias}V_{dd}(mn + 4m\frac{C_{c0}}{C_{in0}}) \end{aligned} \tag{22}$$

where m is the number of outputs. P_{VMM} scales linearly with the number of inputs and outputs, while P_{WTA} scales with the number of outputs only. This is because the common node capacitance scales with the number of WTA outputs as $C_c = mC_{c0}$.

We assume a settling time of 4τ to calculate the computation performed by the classifier. The VMM computation is $m * n$ MAC. The WTA computation is more involved, and is equivalent to solving dynamical equations at the m input nodes and the common node. For an equivalent ODE simulation using Runge-Kutta 4^{th} and 5^{th} order adaptive integrator (RK45), we need approximately 60 MAC per node. Thus the effective computation performed by the classifier can be approximated as $C = (m * n) + 60 * (m + 1)$MAC. The power per unit computation can be calculated as

$$\frac{P * (4\tau)}{C} = \frac{4m(n + 4\frac{C_{c0}}{C_{in0}})C_{in0}U_TV_{dd}}{(m * n) + 60 * (m + 1)} \tag{23}$$

The computing efficiency is plotted in Fig. 30. We assume that $C_{in0} = C_{c0} = 1pF$ for this calculation. For large number of inputs and outputs, the VMM efficiency (which is constant), dominates. For smaller outputs from the classifier, the WTA efficiency dominates.

3.6.3 Temperature Effects

The programmed weights have a direct temperature dependence due to U_T, as seen in (11). In a classifier with a differential VMM implementation, as seen in Fig. 31, it is possible to compensate for temperature effects [97]. To derive the temperature dependence, we first note that the WTA output voltage V_{out} in Fig. 31 is directly proportional to U_T. This is true, whether in the balanced case (gain determined by Early voltage) or in the winning case (gain determined by diode-connected nFET), and only the proportionality constants differ. We use the exponential formulation for Early voltage in the nFET drain current equation given by

$$I_{nfet} = 2I_{th}\frac{W}{L}e^{(\kappa(V_g - V_{T0}) + \sigma V_d)/U_T} \tag{24}$$

where $\sigma = U_T/V_A$.

We determine the current from a single differential cell in terms of a reference temperature T_0.

$$\begin{aligned} I_1 = I_b w^{T_0/T} e^{V_b/U_T}[(1 + \frac{\Delta w}{2w})^{T_0/T}(1 + \frac{x}{2}) \\ + (1 - \frac{\Delta w}{2w})^{T_0/T}(1 - \frac{x}{2})] \end{aligned} \tag{25}$$

which can be approximated, by ignoring higher order terms, as

$$I_1 = 2I_b w^{T_0/T} e^{V_b/U_T}[1 + x_1\frac{\Delta w_1}{4}\frac{T_0}{T}] \tag{26}$$

V_b is the common mode input voltage, x is the differential input normalized to U_T, and w is the bias weight. We assume that the bias weight $w = 1$ and express the

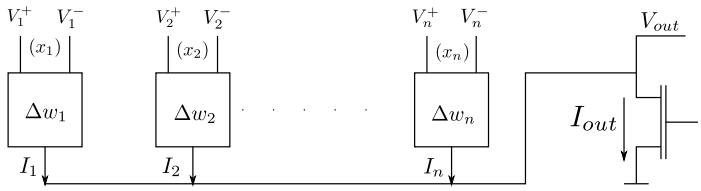


Figure 31: **Temperature dependence:** A classifier with a differential VMM can be compensated for temperature.

total WTA input current as

$$\begin{aligned} I_{out} &= 2I_b e^{V_b/U_T} \sum_{k=1}^{n} [1 + x_k \frac{\Delta w_k}{4} \frac{T_0}{T}] \\ &= 2nI_b e^{V_b/U_T} [1 + (1/n) \sum_{k=1}^{n} x_k \frac{\Delta w_k}{4} \frac{T_0}{T}] \end{aligned} \tag{27}$$

For small increases in temperature, we can assume that V_g remains fixed, resulting in the WTA output voltage

$$V_{out} = \frac{U_T}{\sigma} log(\frac{2nI_b}{I_{th}}) + \frac{V_b}{\sigma} + \frac{U_{T0}}{4n\sigma} \sum_{k=1}^{n} x_k \Delta w_k \tag{28}$$

The WTA output voltage consists of a bias term, which is temperature dependent and the signal term which is temperature independent. We note that the signal term contains x which shows no temperature dependence when the differential input to the VMM scales with temperature.

3.7 Conclusions

Analog classifiers can provide a low-power alternative to DSP techniques and a variety of techniques have been proposed for low-precision applications [32, 111, 113]. One of the drawbacks of using analog is fixed functionality in the classifier. We have presented results from a powerful re-programmable classifier that can implement linear as well as nonlinear decision boundaries. The classifier architecture combines two power efficient circuits to provide an ASP alternative to traditional approaches. The system is extremely compact, allowing scaling to large number of inputs. One of the disadvantages of ASP is fixed functionality. The reconfigurability of the chip allows

programmable weights which enables off-line training, modifying the size and changing the topology of the WTA to generate different behavior. As an extension to this work, we can implement local WTAs and hysteretic WTA for certain applications. We have seen that the VMM+WTA classifier is roughly equal to a 1-layer NN in circuit complexity, but has computing power equivalent to a 2-layer NN. We demonstrate this by implementing classic small-scale nonlinear classification problems.

CHAPTER IV

RECONFIGURABLE NEURON ARRAY WITH PLASTIC SYNAPSES AND PROGRAMMABLE DENDRITES

What if FPGAs were able to take inspiration from biology? One of the greatest drawbacks of FPGAs today is the high power consumption. The advantage of reconfigurability offered by FPGAs comes with the cost of large size, parasitics in the interconnect which in turn limit use in high-frequency applications and the poor power efficiency. An FPGA consists of computational blocks embedded in a global interconnection architecture. The programmable global interconnect, along with local interconnection present in the computational units provides great flexibility and allows quick implementation of many digital systems. A majority of the static power dissipation in FPGAs can be attributed to gate oxide leakage currents in the interconnect, while the dynamic power dissipation can be attributed to the high capacitive load posed by the interconnection fabric. Static memory cells used for programming connectivity and configuration, although optimized for area, contribute to increased size of routing lines which further accentuates the power dissipation issue.

While traditional research on FPGAs have focused on reducing size and improving functionality, there is a great need today to focus on building architectures that are more power efficient. Contrast this to biology, where large networks of neurons exist, performing complex computations with extremely low power consumption. How do biological networks differ in their connectivity and complexity and how can we use that information to build more power efficient FPGAs? Anatomical studies have suggested that in biological networks, most of the connections to neurons are nearest-neighbor type, local connections with very few global connections. Neurons

themselves are complex analog processing units with several state variables and rich dynamics. Further, communication between neurons occur with digital events known as action potentials, but the connectivity and computation in the neurons ensure that the rate at which events occur is about 1 Hz.

This chapter is organized as follows. In Section 4.1, we discuss the chip architecture and the reconfigurable routing. Next, we describe the components of the Neuron block, including the synapse, dendrite and soma elements. We provide measured results characterizing the synapses, soma and summation of inputs over the dendrite. We conclude with the application space for this chip and potential computational efficiency.

4.1 A Neuromorphic FPGA

Recent efforts at emulating neuron activity in the visual cortex have shown that the main challenges are power consumption and scalability [2, 67]. Several researchers have built neuron arrays in the past [47, 114] and their approach has been to create a compact array of neurons with interconnectivity handled by a digital interface, since neuron outputs can be considered digital events. Address Event Representation (AER), developed by Mahowald and Sivilotti is a commonly used digital communication standard for hardware neuron chips in which an event from a neuron is represented by the address of that particular neuron on the address bus [65]. AER transceivers on-chip can direct inputs to specific neurons, collect outputs from neurons and can "program" connections between neurons using SRAM to store connections.

The AER interface is very useful for direct digital communication to the chip, as well as interfacing with a variety of neuromorphic sensors which also provide digital outputs, thereby making it possible to build larger systems that do high-level processing. However, while this approach ensures all-to-all connectivity, it also accounts for the majority of power dissipation in the neuron chips. Also, the power consumption

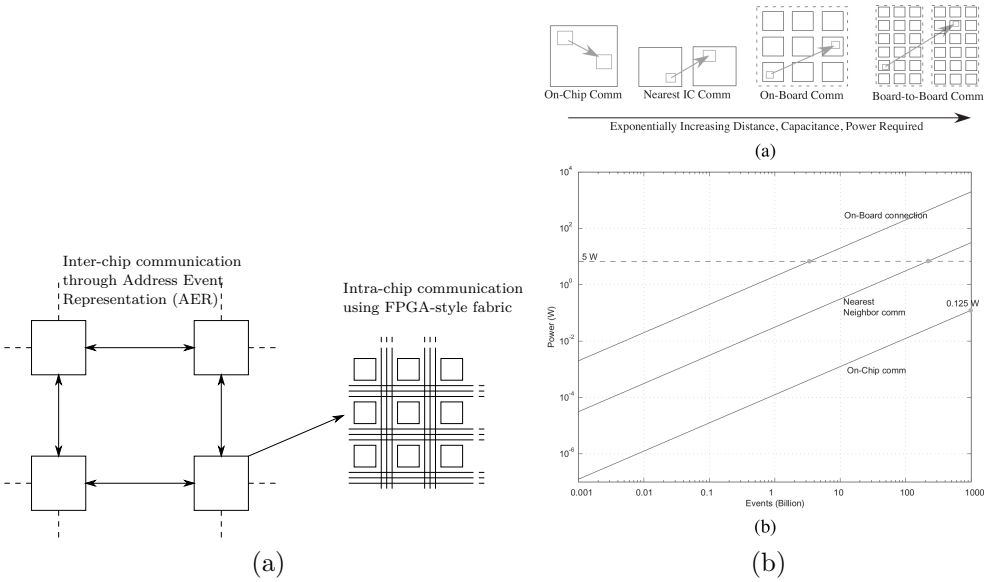


Figure 32: **Chip Design Philosophy:** Large scale neuron arrays can quickly become power hungry. Using the knowledge base that motivated FPGA designs and architectures and learning from physiological studies that suggest that connections between neural computation centers remain largely local, a system design flow is developed. Single neurons within a chip communicate using local routing, using local memories embedded in FPGA style architectures. Off-chip communication is more communication and hence kept sparse, handled by well known digital AER interfaces.

does not scale linearly with the number of neurons.

The power consumption due to memory access and the bottleneck of intra-chip neuron connectivity through the AER interface can be reduced by using FPGA-style routing for specifying interconnections. This scheme decreases the number of events transmitted on the digital bus, reducing the load on the AER infrastructure. This also mirrors network topologies in biology, where dense connectivity is observed between neurons close to each other and connections between neurons far away remain sparse. By using the programmable routing fabric for local connectivity and using the AER transceiver for off-chip communication as seen in Fig. 32a, overall power consumption can be reduced significantly while retaining the advantages of having a digital interface. We present a chip designed based on these principles and fabricated in the 0.35μm process, shown in Fig. 33b.

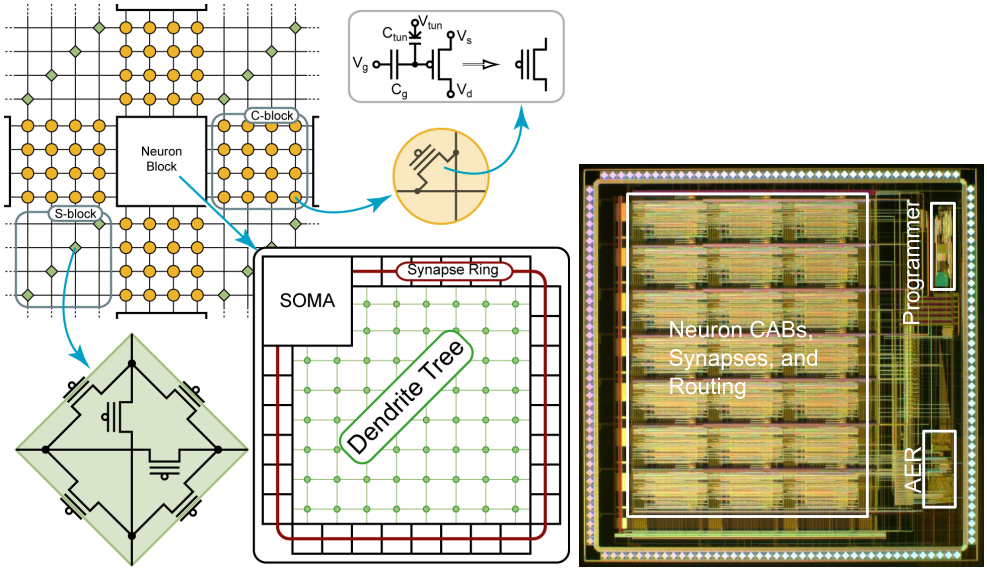


Figure 33: **Chip Architecture:** The Neuron2 chip consists of neuron cells embedded in a typical FPGA routing fabric. The Neuron I/O interface with the routing at the C-Blocks through programmable floating-gate switches. The tracks are segmented for allowing faster event transmission and maximizing utilization. The tracks are routed at the S-Blocks, where each node consists of 6 switches. The neuron cell has synaptic inputs, programmable dendrites with active channels that aggregate inputs into the soma block.

4.1.1 Chip Architecture

The chip presented in this work (**Neuron2**) consists of 21 programmable neuron cells that are biophysically inspired. Each neuron cell consists of several synaptic inputs, capability for specifying a 2D dendritic structure with active channels, programmable channels and a 2D soft winner-take-all network. The synaptic inputs to the neuron cell and the neuron output can be routed on the programmable fabric. Some of the routing lines are connected to the AER interface for off-chip communication. Fig 33a depicts the neuron tile, consisting of a neuron cell and associated routing. A summary of key chip parameters is provided in Table 3.

4.1.2 Global Interconnect

The neuron cells are equivalent to the computational logic blocks(CLBs) in a reconfigurable Manhattan-style interconnect architecture. The Manhattan-style interconnection offers advantages of reducing parasitic capacitance while allowing greater utilization of routing lines. This is achieved by segmenting the routing fabric into lines spanning the width of one neuron cell. The routing architecture consists of the C-Block and S-Block segments. The C-Block is used for making connections between the Neuron I/O and the routing tracks. The S-Block allows connections between the track segments. All the switches in the routing fabric consist of floating gate transistors that may be programmed, much like EEPROM memories.

The total number of neuron cells and their connectivity affects the choice of the number of tracks in the global routing, which impacts the size of the neuron array. There is a trade-off in the number of tracks (and hence, total size) and arbitrary connectivity between the neuron cells. One approach to further improve scaling and reduce size is to force connectivity between neighbor and nearest-neighbor cells, with tracks being used only for other connections. In the Neuron2 chip however, 11 global tracks are used and connectivity is not forced since the chip only contained 21 neuron cells.

The neuron cells themselves are reconfigurable blocks of models of neural computation(Neuron CAB), that accept digital events from other neurons as inputs. The interface to the neuron block consists of synapses which process digital events and convert them into analog signals. In the following section, the parts of the neuron CAB are discussed in greater detail.

4.2 Silicon Neuron Model

The neuron cells themselves are reconfigurable blocks of models of neural computation (Neuron CAB), that accept digital events from other neurons as inputs. The interface

to the neuron block consists of synapses which process digital events and convert them into analog signals. In the following section, the parts of the neuron CAB are discussed in greater detail.

Table 3: Key parameters of the Neuron2 Chip.

Parameter	Value
Size	25 mm^2
Neurons	21
Synapses	588
Dendritic channels	315
Programmable Parameters	1932

4.2.1 Silicon Synapses

Neurons provide inputs to other neurons through synapses. A fact that provides some perspective of the neural structure is that there is an estimated 1 billion synapses per cubic millimeter of cortical gray matter [[53]]. Based on this calculation, there are about 10^{15} synapses in a typical human brain! Synapses may be electrical (or gap junctions) or chemical synapses, which are more common. Some connections between neurons may be direct electrical connections or gap junctions, which can be modeled as a conductance between two neurons. Chemical synapses on the other hand are involved in the activation of chemical pathways that cause changes in the post-synaptic neuron.

Synapses are usually between the axon of the pre-synaptic neuron and the dendrites of the post-synaptic neuron. In some cases, the pre- and the post-synaptic neuron may be one and the same, as in cortical inhibitory neurons present in the Winner-Take-All structure. Pre-synaptic terminals contain sacks or "vesicles" containing neuro-transmitters. An action potential at the pre-synaptic terminal causes a Calcium influx, which causes the vesicles to fuse with the pre-synaptic membrane at specific sites. Neuro-transmitters are released into the synaptic cleft and diffuse accross the separation, which is usually just a few nanometres wide and bind to the

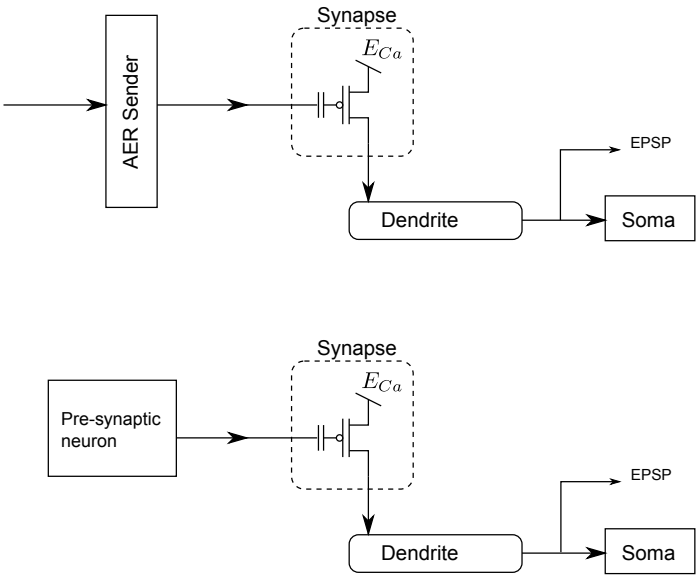


Figure 34: Inputs to synapses may be generated by other neurons or externally, by the AER sender.

post-synaptic receptors. These are responsible for selective opening of channels (e.g. Na), causing an increase in post-synaptic membrane potential.

An input at the pre-synaptic terminal causes a change at the post-synaptic terminal. Synapses may be excitatory or inhibitory. Excitatory synapses cause the post-synaptic terminal(membrane) to depolarize and inhibitory synapses cause the post-synaptic terminal to hyperpolarize. The synapse circuit model and dynamics are discussed in greater detail in Chapter 5.

The Neuron2 chip has 28 synaptic inputs, of which 20 synaptic weights can be modified based on network activity in run time based on the STDP learning rule [87]. Each synapse itself is a floating gate transistor whose weight can be precisely programmed. The other 8 synapses can be configured as excitatory or inhibitory synapses. Inputs to neurons may be generated by other neurons or externally, through the AER sender. To elicit an appropriate EPSP, a digital input event undergoes waveform shaping [87] to generate a triangle waveform shown in Fig. 39a.

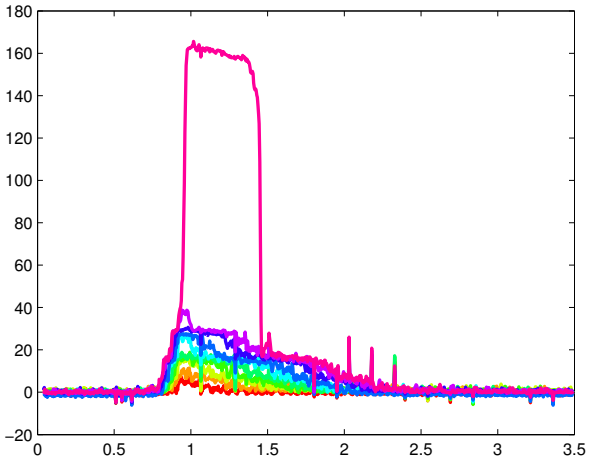
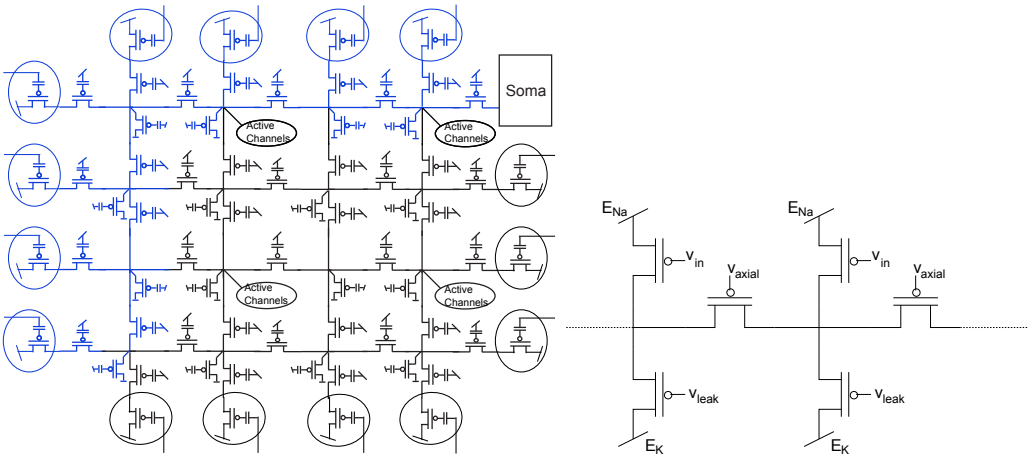


Figure 35: EPSP on soma in response to a synaptic input. Inputs events to neurons are converted into a triangle waveform with programmable duration and slopes.

4.2.2 Dendritic modeling

The role played by dendrites in neural computation remains a controversial issue. Dendrites are widely believed to merely aggregate inputs received from synapses and do passive filtering. This hypothesis leads to the treatment of dendrites as wires and the reduction of the compartmental neuron model to the point neuron model. Wilfrid Rall proposed a model for dendrites and hypothesized that channels present at dendrites may result in some nonlinear behavior [85, 99]. In recent literature, evidence has been presented which indicates that dendrites may play a greater role in the processing done in each neuron [40,61]. In this chip, we choose to model dendrites in the cell to study possible computation performed by them.

Each neuron cell in the Neuron2 chip features a fully reconfigurable 2-D dendritic network, supporting arbitrary arborization. The dendritic fabric is modeled as in [76], with conductances set using floating gate transistors which allows tuning of diffusion constants along the line. Every other node in the dendrite also contains programmable active channels (a band pass channel and a low pass channel).



(a) (b)

Figure 36: **Dendritic Structure:** (a) Depiction of the neuron cell structure, with arbitrary dendritic structure capability. Dendrites are also interspersed with active channels to model the nonlinear behavior observed in biology. In blue, we show a 8-tap dendritic line programmed on the neuron. (b) Model of a silicon dendrite. The axial and leak conductances are set by the horizontal and vertical transistors respectively.

4.2.3 Neuron Soma

The analog current output from the dendrite feeds into a soma, which is a configurable block consisting of nonlinear channels and local soft WTA. The channels consist of programmable Bandpass channel (Na^+ model), a Low pass channel (K^+ model) and a leak channel that result in a digital output or event from the neuron. The channels are modeled as discussed in [23]. The schematics of the channels are shown in Fig 38. An OTA configured as a buffer is used to observe the spiking dynamics of the neuron and for debugging purposes. An OTA based comparator is used to generate digital events from membrane spikes. The threshold is set as a global parameter but the OTA inputs are floating gates which allow cancellation of offsets due to mismatch in the input pair of the OTA as well as the variability in spiking thresholds.

Also included in the soma is circuitry that enables Spike Time Dependent plasticity in the synapses. These circuits involve modifying the tunneling and drain terminals of the synapse FG transistor during "run time" [87]. The thresholded neuron output

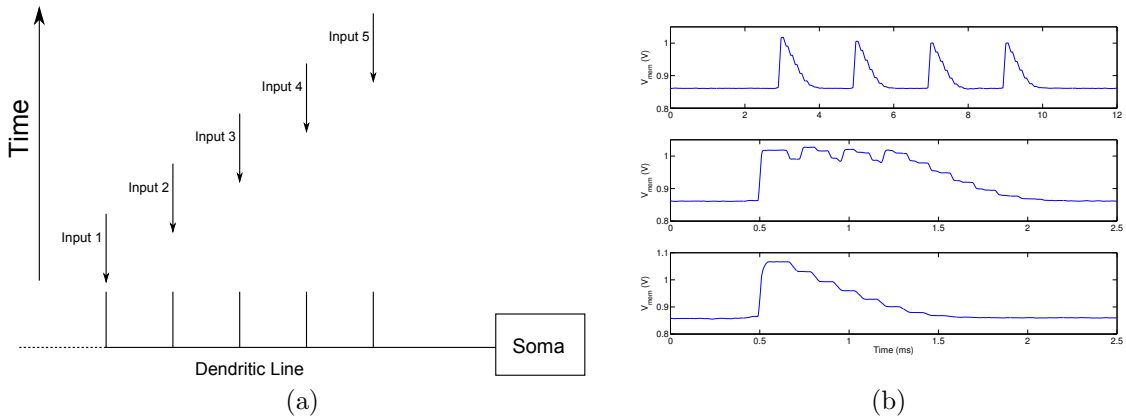


Figure 37: **Summation of synaptic inputs on the dendrite:** Summation of synaptic inputs on a 4-tap dendrite line: synaptic inputs aggregate over the dendrite, eliciting an EPSP with higher peak voltage for inputs occurring simultaneously.

can be routed to synapse inputs of other neurons, but since each synapse would require a waveform shaping circuit to generate post-synaptic currents similar to biology, redundancy is avoided by routing the pre-synaptic shaped waveform over the routing. Thus, the FPGA-style routing is not really used for routing digital events, but for analog waveforms. Each neuron event generates various timing waveforms, shown in Fig. 39, with a triangle waveform to generate inputs to synapses on afferent neurons, and STDP control pulses that govern learning for synapses on the active neuron.

4.2.4 AER

The Neuron2 chip contains an AER interface to for external communication. The interface has been synthesized using Cadence tools from a high level description in Verilog. The AER transmitter which takes events from neurons and transmits addresses for the events, takes rising-edge triggered inputs and latches all of the events in a given time period into an array of N flipflops. The latched structure of stored events then converts each event, in turn, to an address on the output bus. The AER receiver module takes input event addresses and communicates them to the array. It

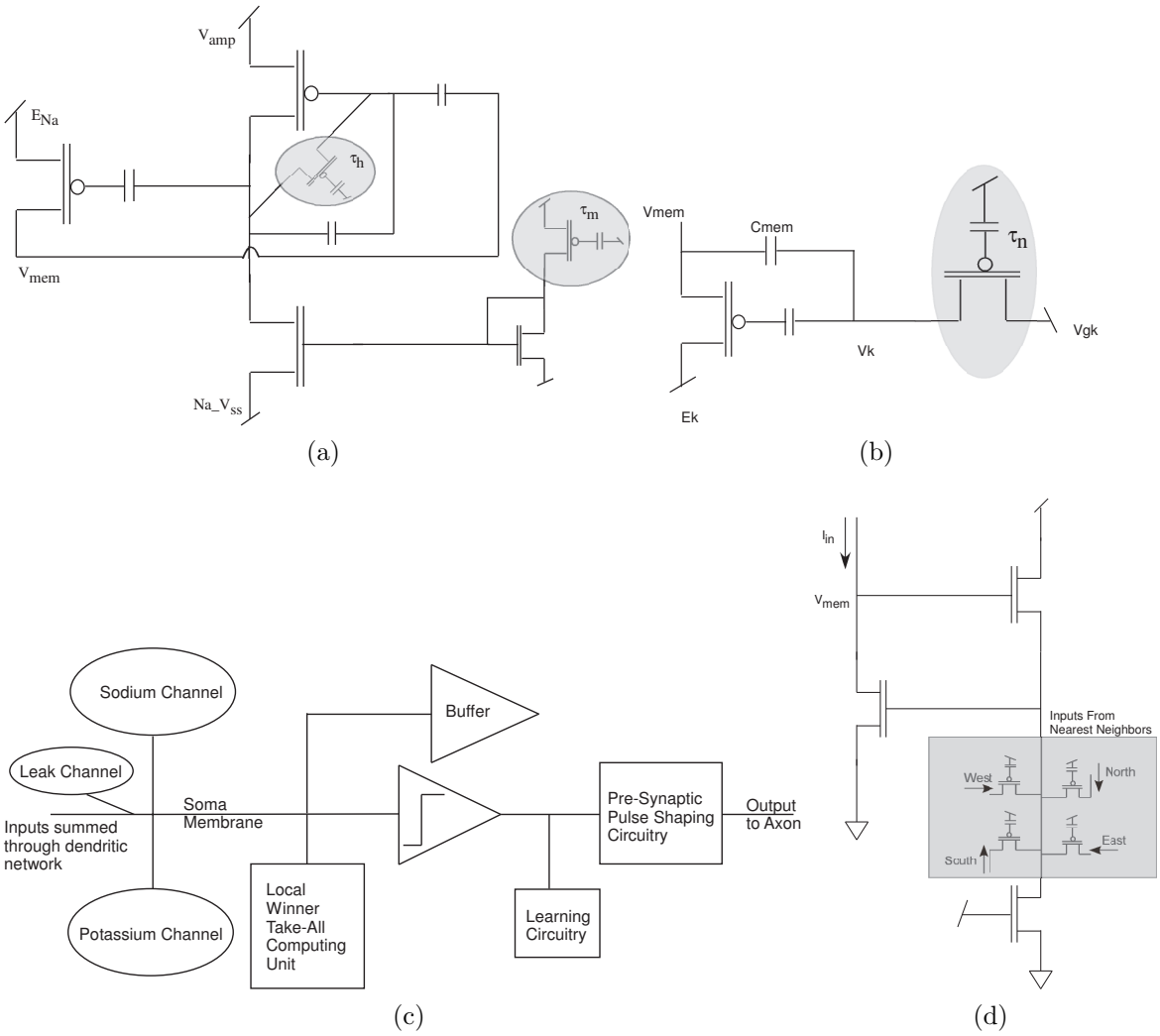
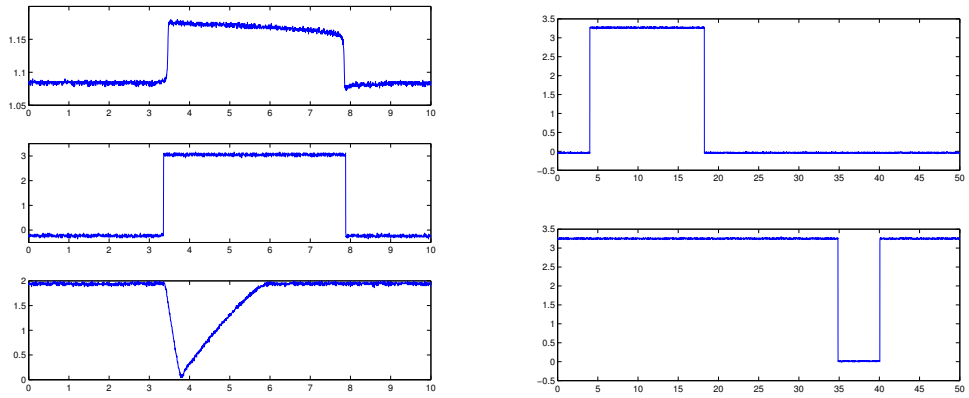


Figure 38: **Detailed view of Configurable Soma:** (a)Programmable structure modeling the Na channel. The Na channel transistor itself is a floating-gate device to allow ease of biasing. The activation and inactivation time constants are set by two other floating-gate devices, marked in grey. (b) Structure modeling K channel, slow activation time constant set by a floating gate device shown. (c) Besides the channels, the soma also consists of a local WTA unit, a thresholder to generate digital events, a buffer for debugging and learning circuitry that can be enabled. The axonal output is actually a shaped pre-synaptic waveform. (d) A local WTA computing unit can accept inputs from its neighbors in all four directions. This will greatly reduce the event rate in our networks besides modeling mechanisms seen in interneurons in the visual cortex.



(a) (b)

Figure 39: Timing waveforms generated as a result of a neuron spike: triangle waveform routed as neuron output, drain control waveforms with programmable delay and pulse width for injection control and tunnel waveform for tunnel control. The drain and tunnel waveforms can be enabled for allowing learning in the synapses.

is functionally a digital decoder followed by a circuit to hold the event for a particular duration to be input into the pre-synaptic waveform shaper circuit. In this chip, individual neurons are not addressed, instead AER receiver is used to generate events that can be routed to individual neurons, or address multiple synapses on the same neuron. The AER sender is used to collect events from specific global routing lines that can have events routed onto them from individual neurons.

4.3 Dendritic computation

The computation in dendrites is highly debated, particularly given the complexity and computation richness available here. In many modeling efforts, the dendrite is approximated to be a wire, greatly simplifying the resulting network and enabling a system that is tractable by a range of computational principles. However, given recent results that indicate efficient computational models using dendrites, we investigate dendritic computation in speech processing applications.

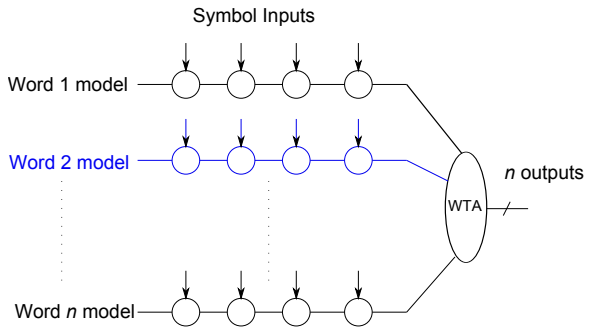


Figure 40: **Word detector:** Block diagram of dendritic lines being used to detect words

Neurons with a basic dendritic structure can implement a word-spotting algorithm, a key engineering approach for many classifier applications. The block diagram of a word-spotting network using dendrites is shown in Fig. 40. Each dendritic line represents a word model, with the states extracted after an HMM training sequence. The synaptic inputs into each line represent the probability of inputs to that word model, and the dendritic line computes the probability of the word occurrence. We exploit temporal summation and directional selectivity properties of the dendrite line. These properties ensure that the dendrite line responds with the largest magnitude EPSP for the "preferred" direction of inputs, while not responding to the "null" direction. The "preferred" direction corresponds to the inputs on the dendrite line appearing in sequence from the distal to the proximal end, while the "null" direction corresponds to inputs in sequence from the proximal to the distal end.

The dendrite line tuned to exhibit directional selectivity is shown in Fig. 41. The response of the dendrite in the "null" direction is similar to when only the proximal input is applied to the line. This can be achieved by increasing the axial conductance and decreasing the leak conductance when moving from distal to proximal end of the soma. This effect is similar to the increasing diameter of the dendrite towards the soma.

The dendrite shows directional selectivity over a range of delays that are in the order of the signal propagation delay on the dendrite. Fig. 42 plots the difference

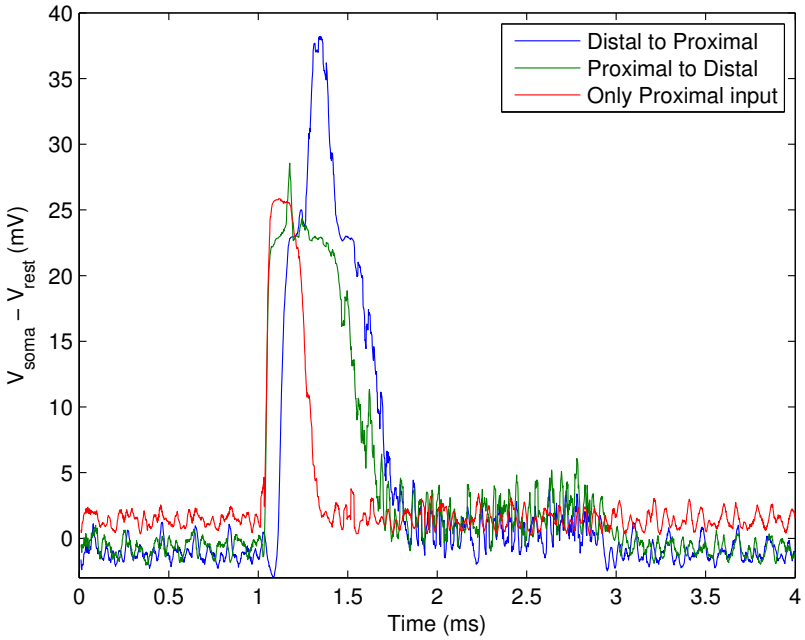


Figure 41: **Directional Selectivity:** A dendritic line is directionally selective to a temporal sequence of inputs arriving from the distal end to the proximal end when having an increasing diameter towards the soma. The effect of the increasing diameter can be modeled by increasing the axial conductance and decreasing the leak conductance.

between the voltages at the soma for activation in the "preferred" and "null" directions. This line is sensitive to inputs that have a 40ms delay between them. When the delay between inputs is very small, the

4.4 Conclusions

A novel neuron chip architecture that shows promise in building large-scale neuron arrays has been described. The architecture minimizes use of power-hungry external digital memories for on-chip connectivity by leveraging local memory present in the floating-gate routing. The neurons in this chip are not considered point-neurons and a programmable dendrite structure with active channels are included. The synapses in this chip support learning through STDP.

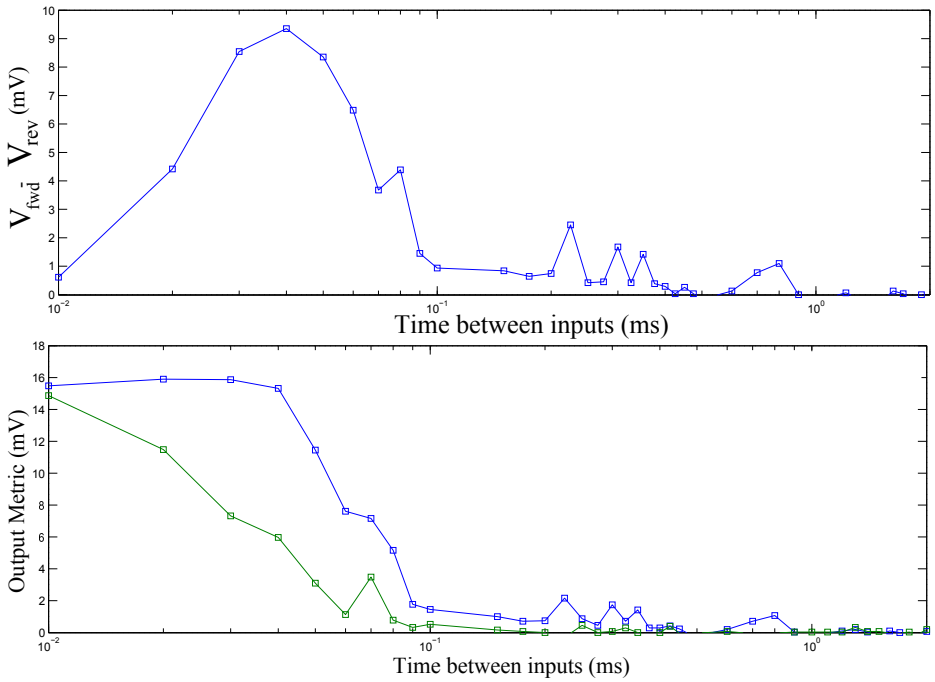


Figure 42: **Characterization of line sensitivity to delay between inputs:** The directionally selective dendrite line responds to a limited range of delay between the inputs, which is related to the delay due to the dendrite itself.

The goal of neuromorphic engineering is to develop biologically inspired engineering solutions that offer a significant computing and power advantage over mainstream digital approaches. It is therefore essential that while building large scale neuron arrays, we take a low-power approach. We intend to use this neuromorphic chip for implementing a low-power word detector, discussed in [24]. This system takes speech symbols detected by a front-end processing unit and detects sequences to identify words. This is an alternative implementation to DSP-based speech recognition engines, and shows promise in being more computationally efficient. In addition, this system implementation requires dendritic lines performing sequencing and can be an interesting hypothesis for the role of dendrites in neural computation.

CHAPTER V

FLOATING GATE SYNAPSES WITH SPIKE TIME DEPENDENT PLASTICITY

Currently there is a great interest in implementing large scale biological networks in silicon to provide a substrate for performing hardware simulations of neural networks. As part of this effort, it is essential to model the learning processes inherent in biological synapses. Many of these learning processes are forms of Hebbian learning observed in biology, and while they are often interpreted as changes in synaptic weight based on correlations between mean firing rates of pre- and post-synaptic neurons, [9] and [68] describe synaptic learning rules that are governed by precise timing differences between pre- and post-synaptic spike times. As shown in Fig. 43, when a post-synaptic spike follows shortly after a pre-synaptic spike, the synaptic weight is increased, but when the order of spikes is reversed, the weight decreases.

Over a decade ago, the concept of a single transistor learning synapse (STLS) with experimental results was proposed, similar to a modern EEPROM cell that could simultaneously store a weight in a non-volatile manner and compute a product of the input by the weight and adapt the weight based on signals in the array [34]. The specifics of adaptation mechanisms for an individual device were described but not well defined in terms of usable adaptation algorithms. Since then, there have been efforts in adaptive amplifiers [39], adaptive filters [36] as well as development of techniques enabling least means squared (LMS) adaptive filters [37].

In biology, synapses strengthen through Long-Term Potentiation (LTP) in which chemical and morphological changes are made to improve signal transduction from

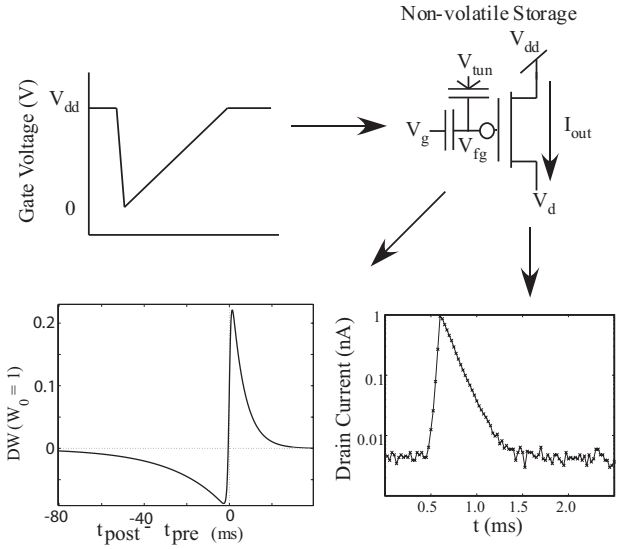


Figure 43: **Single Transistor Learning Synapse:** stores a "weight" in a non-volatile manner and computes a Post-Synaptic Current(PSC) with a triangle wave input generated by a pre-synaptic computation block. The synapse also implements a weight update rule which depends on the time difference between pre- and post-synaptic spike times.

the pre-synaptic to the post-synaptic cell [1]. After LTP, the post-synaptic potential(PSP) is much stronger than it was before LTP. Conversely, long-term depression (LTD) decreases synaptic strength such that the PSP becomes weaker.

In this discussion we present a STLS device and architecture that stores a weight in a non-volatile manner, computes a biological post-synaptic potential (PSP) and demonstrates biological synapse learning rules such as LTP, LTD and spike-time dependent plasticity (STDP). In order to explore the intrinsic learning rules that can be found with the STLS, we used variable inputs to the gate, tunneling node and the drain. Inspired by the experiments of Bliss and Lømo [10], the initial PSP was measured. A particular sequence of input and output spikes was repeated several times and the final PSP was noted. Fig. 44 shows the excitatory post-synaptic currents (EPSC) obtained after 10 pairings of the pre- and post-synaptic spikes. The weight increases when the post-synaptic spikes follow the pre-synaptic spikes and decreases when the order is reversed, as seen from the amplitude of post-synaptic currents.

This current is measured with a triangle waveform at the input generated by the pre-synaptic computation block. Our synapse was fabricated in a widely available 0.35μm double-poly CMOS process. The size of the synapse transistor in this work is 1.8μm/0.6μm. Initial results from the STLS were presented in [88]. In Section

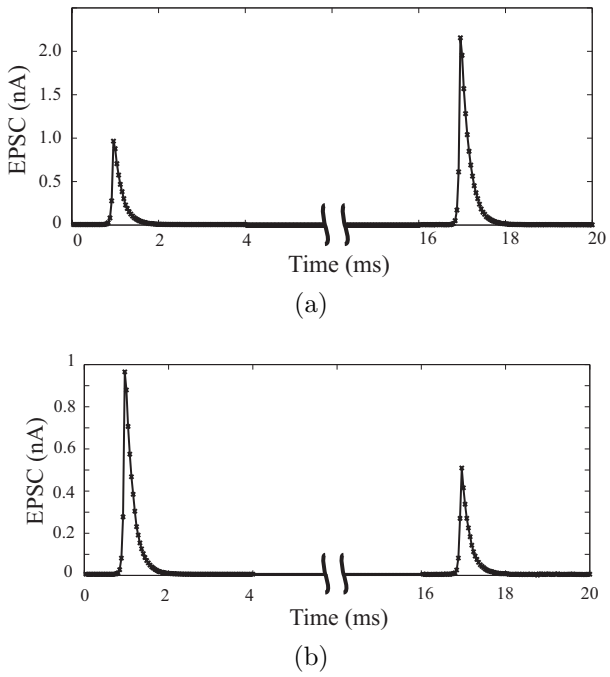


Figure 44: **EPSC before and after learning:** Similar to the Bliss and Lømo experiments, we record EPSCs before and after learning events that are activated by post-synaptic spikes. The order of pre- and post-synaptic spikes determine whether the synapse is "potentiated" or "depressed". (a) EPSC after 10 pre-post pairings. (b) EPSC after 10 post-pre pairings.

II, the basics of transistor learning synapses is presented. Section III discusses the algorithm used to implement LTP and LTD. A mathematical model for the learning rule implemented in the STLS is derived in Section IV. In Section V, data from STDP experiments is presented.

5.1 Basics of Transistor Learning Synapses

Fig. 45 shows one configuration of single transistor learning synapses (tunneling junctions not drawn for clarity). In this chapter, we present measured results from

a single synapse device and propose an architecture for n neurons and an $n \times n$ synapse array that supports all-to-all connectivity. A very high density of synapses is obtained, and complex circuitry placed near the periphery can be shared among elements in the array. The pre-synaptic computation circuitry is placed to the left of the array and fed into all synapses afferent to the neuron in that row. The output from all synapses in one column feed into an axon block. The control for the learning circuitry is also placed in the bottom of the array. Fig. 45 shows the basic approach for the synaptic learning mechanism. A combination of hot-electron injection and Fowler-Nordheim tunneling is used for synaptic weight modification. The learning mechanism is only enabled when an event on a post-synaptic neuron occurs. The synapse is turned into "PROGRAM" mode for a short time, allowing the weight to adapt based on other signals in the array and then reset into "RUN" mode. This approach allows a wide set of potential learning rules to be implemented.

5.1.1 Feed Forward Synapse Computation

In this subsection, we discuss the feed-forward computation of primarily excitatory synapses and the setup for learning. The terminals of the floating gate transistor shown in Fig. 43 are the gate voltage (V_g), drain voltage (V_d), tunneling voltage (V_{tun}) and floating gate voltage (V_{fg}). For this discussion, we assume that the source and well of the transistor are tied to V_{dd}. The pre-synaptic computation block must provide the necessary channel gating voltage to the transistor synapse to get the PSP described in Fig. 43. The synaptic current as a result of an input to a biological synapse is given by [54].

$$I_{syn} \propto te^{-t/\tau_{rise}} \tag{29}$$

where τ_{rise} is typically on the order of 0.1 ms. The exponential nature of the relationship between the gate voltage and drain current of the MOS transistor lends itself to the implementation of a synapse-type structure. Hence the gating voltage to the

synapse has to be a triangle waveform that decreases from its high resting value. The current at the highest gate voltage is nearly zero, within the source and drain leakage currents. The fast decreasing part of the input with slope s_1 results in a quick rise of the synaptic current while the slowly increasing input with slope s_2 determines the exponential decay in the output current. In order to generate EPSCs similar to biology, we apply a triangular wave with unequal slopes at the input. A current starved inverter structure [29] is used to convert the action-potential like digital pulse into a triangular waveform, with the slopes precisely controlled using floating gate biases. The strength of the synapse or the "weight" is part of the proportionality constant of (29). The pre-synaptic triangle waveform also assists in shaping the learning rule implemented, as discussed in Sections III and IV.

Hot-electron injection is used to add electrons on the floating gate node, resulting in more current from the transistor and increasing the weight of the synapse. Electron tunneling removes electrons from the floating-gate node, reducing the drain current of the transistor thus decreasing the synaptic weight. The amount of injection and tunneling depend on the current through the device and the field across the tunneling oxide respectively, which again depends on the gate voltage. Since the instantaneous gate voltage depends on the time since the pre-synaptic spike (marking the start of the triangular waveform), injection and tunneling currents also depend on the time difference between pre and post-synaptic spike times. We define the drain current of a sub-threshold saturated floating-gate pFET as

$$I_d = I_{bias} w e^{-\triangle V_g / V_{gc}} \tag{30}$$

where I_{bias} is the quiescent bias current through the device, V_{gc} is a constant determined by the capacitive coupling between the gate and the surface potential of the channel, and $\triangle V_g$ is the change in gate voltage. The weight of the synapse w is

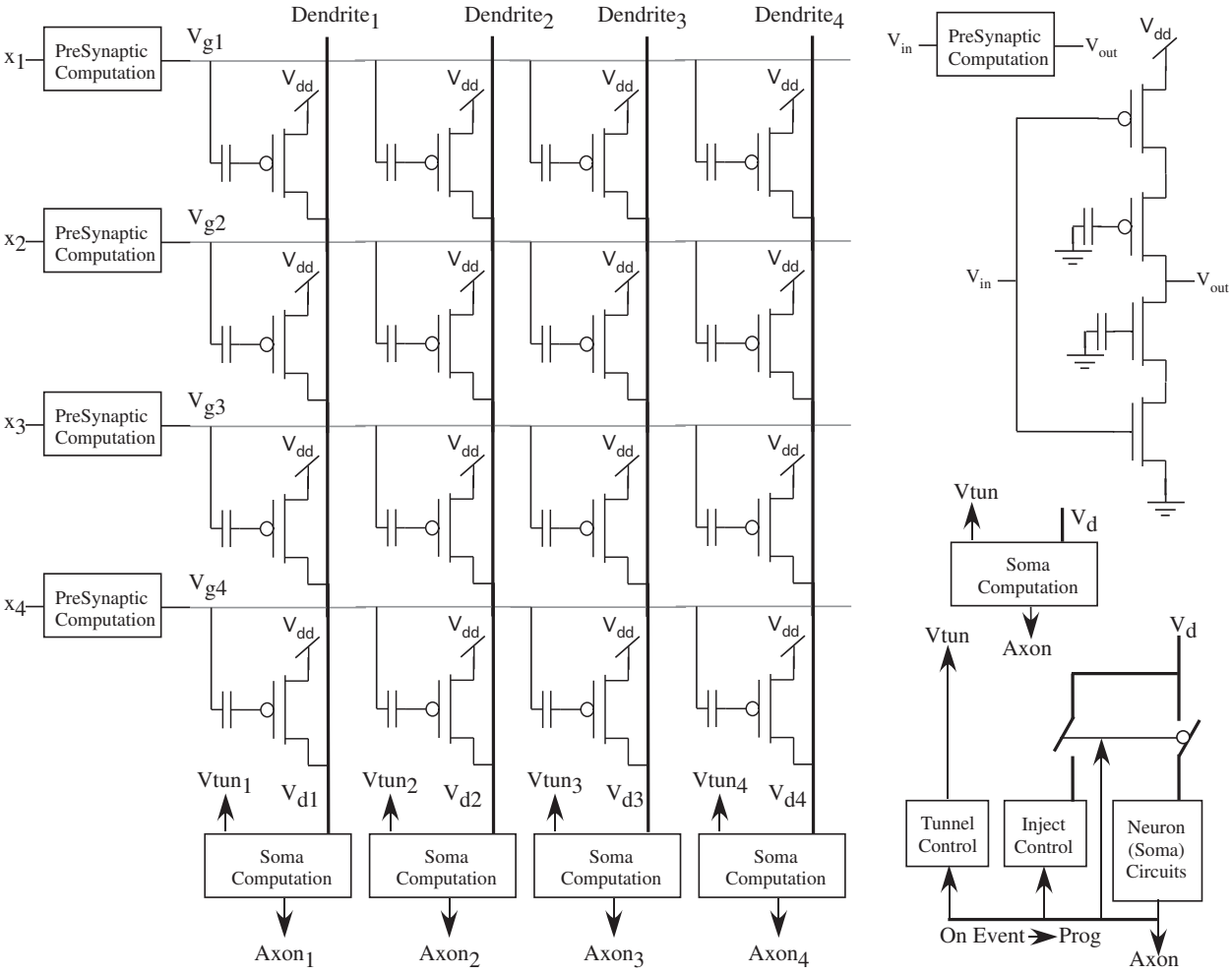


Figure 45: **Array for Learning Synapses:** An array of synapses that allow all-to-all connectivity between neurons results in a very high density of synapses per neuron. This architecture also allows the pre-synaptic computation block to be shared across an entire row. Tunneling lines are not shown, but are shared along the column of the array. When an axon event occurs, the entire array is turned into "PROGRAM" mode, and the Inject and Tunnel control blocks are activated. When an input event occurs (pre-synaptic spike), the pre-synaptic computation block produces a triangular waveform which feeds into all the synapses along that row.

related to the floating-gate voltage V_{fg} by

$$w \propto e^{-V_{fg}} \tag{31}$$

To build the synapse structure, we need a floating-gate transistor and the required pre-synaptic circuitry. We previously presented some initial data on the synapse feed-forward function, including schemes for excitatory, inhibitory and NMDA synapses in an older CMOS process [23]. The transistor and resulting power supply model the post-synaptic channel population. This approach is based on modeling a biological channel with the channel of a MOSFET device in sub-threshold regime of operation [29]. The floating gate structure allows us to store a weight value, which is proportional to the amplitude of the post-synaptic potential. Therefore, the pre-synaptic computation must create a waveform specified by equating currents in (29) and (30) as

$$te^{-t/\tau_{rise}} \propto I_{bias} we^{-\triangle V_g/V_{gc}} \tag{32}$$

where the solution for $\triangle V_g$ is approximated by a linear change in voltage with time. For the exponentially falling tail of the response, (32) simplifies to

$$\triangle V_g = \frac{V_{gc}}{\tau_{rise}} t = s_2 t. \tag{33}$$

where $s_2 > 0$, and we have a linearly increasing gate input with time.

5.1.2 Synaptic Weight Updates

We find the update equation for synaptic weight by considering injection and tunneling currents. The reader is referred to [39] for a detailed discussion on the models for injection and tunneling mechanisms. The simplified expressions are

$$\begin{aligned} I_{inj} &\propto w^{\alpha} e^{-\alpha \triangle V_g/V_{gc}} \\ I_{tun} &\propto w^{\beta} e^{\triangle V_{tun}/V_{ox}} e^{-\beta \triangle V_g/V_{gc}} \end{aligned} \tag{34}$$

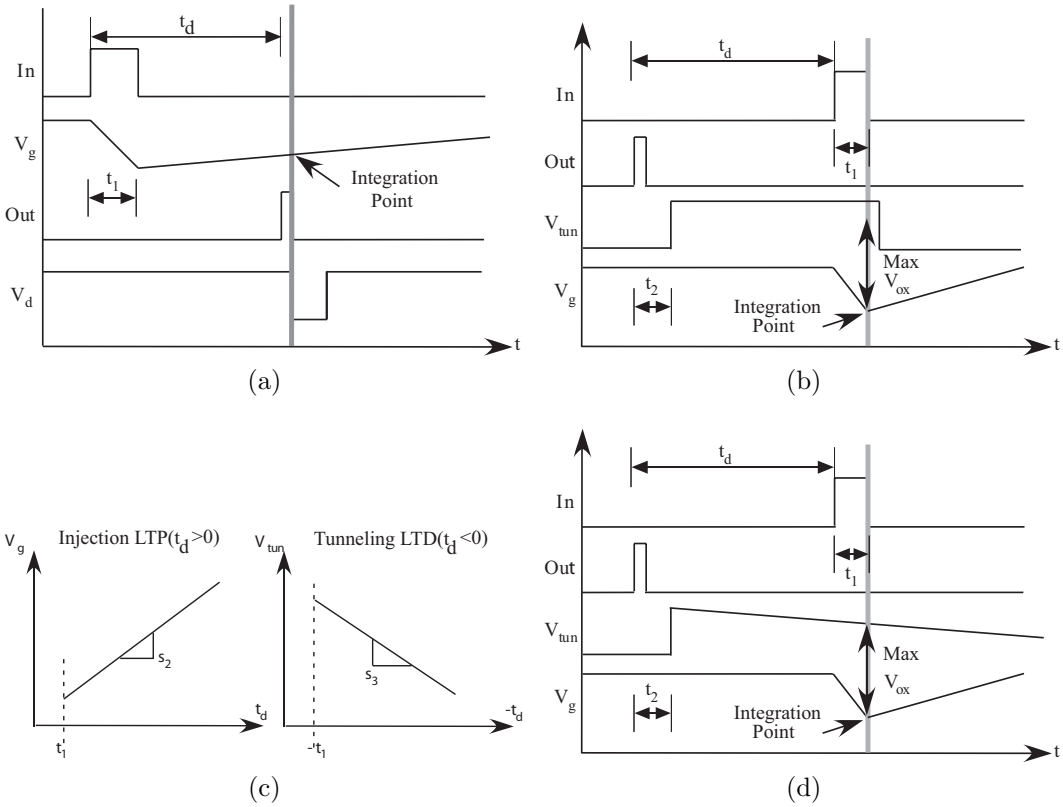


Figure 46: **Timing Diagram for LTP and LTD:** (a) LTP timing rule: When an output event occurs, the drain of the synapse device is pulsed. (b) LTD timing rule with fixed tunnel pulse: At an output event, the tunnel line for the column is pulsed for a fixed duration. (c) Timing relationship for LTP and LTD : We see a linear relationship between gate voltage and positive time delay for LTP and a linear relationship between tunneling voltage and negative time delay for LTD. (d) Modified LTD timing rule: constant tunnel pulse replaced with linearly decreasing tunnel voltage.

where α, β and V_{ox} are process dependent parameters that we define in Section IV. Note that the currents are exponentially dependent on the gate and tunneling voltages. We make the assumption that injection and tunneling currents do not change much in the region of interest. The synaptic weight is a function of the charge on the floating node and hence, using (34), we can write a weight update equation for the synapse as

$$\triangle w \propto T(I_{inj} - I_{tun}) \tag{35}$$

For the LTP portion of the curve, as seen in Fig. 43, we require the change in weight to exponentially decay with increasing time delay between input and output spikes. From (34) and (35), the gate voltage versus t_d, $t_d > 0$ can be directly related to LTP. For an exponential decrease in $\triangle w$ with t_d, a linear change in V_g with a positive slope is required. A similar change in V_g also results in an exponential decrease in $-\triangle w$ with negative t_d. We can also relate the tunneling voltage versus t_d, $t_d < 0$ directly to the LTD portion of the curve. An exponential decrease in $-\triangle w$ with t_d requires a linear change in V_{tun} with a negative slope. These concepts are illustrated in Fig. 46c.

5.2 Learning Algorithm

Although the groundbreaking work which discovered LTP was completed decades ago [10], the full understanding of all of the mechanisms have not been codified. Furthermore, LTD is even less understood. Our system mimics the overall effect of LTP and LTD without relying upon these unknown characteristics. Previously in Section II, we described how to modify the synaptic weight on an STLS. Here in Section III, we will use this knowledge to implement a form of biologically plausible LTP and LTD.

5.2.1 LTP Learning Algorithm

Fig. 46a shows the timing for injection required for the LTP rule and will extend to STDP when combined with tunneling. As seen in Fig. 46a, t_d is the time delay from the start of the input (pre-synaptic spike) to the output (soma/post-synaptic spike) generated. The learning algorithm consists of an injection pulse when the output spike occurs. For simplicity of analysis, we assume without loss of generality that injection occurs immediately following an output spike without delay, however it is possible to modify the learning algorithm by delaying the injection phase relative to when the output spike occurs. When $t_d > 0$, the drain pulse occurs when the gate

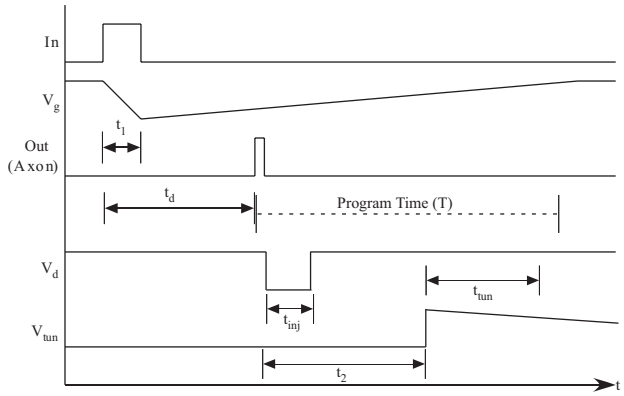


Figure 47: **Timing of the Programming algorithm for the STDP learning rule:** the algorithm shown here uses a linearly decreasing tunnel voltage.

voltage is lower than its quiescent value due to the pre-synaptic computation block, resulting in non-zero injection of the floating gate. For values of t_d larger than the duration of the pre-synaptic waveform, the gate voltage has reached is quiescent value again by the time the drain pulse occurs. The floating gate is programmed such that there is no current through the device when the gate is at its quiescent value, hence the injection phase for these values of t_d results in no change in weight. Similarly, when $t_d < 0$, the optimal conditions for injection are not achieved and there is no change in weight of the synapse. Thus, since the voltage at the gate input of the synapse at the time of the injection pulse can be related to t_d and the slopes of the pre-synaptic gate waveform, from (34), injection current is exponentially related to the t_d and the slope of pre-synaptic gate waveform.

5.2.2 LTD Learning Algorithm

Fig. 46b shows the tunneling timing diagram required for the LTD rule, which when combined with the injection phase, can be extended to the STDP rule. As seen in Fig. 46b, t_d is the time delay from the start of the output (soma) spike to the input (pre-synaptic) spike.

The LTD learning algorithm itself consists of pulsing the tunneling input to the synapse after a post-synaptic spike occurs. During the tunnel pulse, the tunneling

current can be estimated using (34). We can achieve a timing dependence for the LTD model in two ways : we could use the slope of the pre-synaptic gate waveform, as in the LTP case, or we can introduce a timing dependence for the tunneling voltage as described in Fig. 46d. The former leads to more symmetric LTP and LTD learning rules, while we can introduce an asymmetry that is widely observed in biological synapses [9] using a transient on the tunneling voltage. From (34), for a fixed tunnel pulse as shown in Fig. 46b, the tunnel current is only dependent on the change in gate voltage which in turn depends on t_d and the slope of the pre-synaptic gate waveform. When a learning algorithm described in Fig. 46d is used, the tunnel current also depends exponentially on the change in tunnel voltage.

We first consider the case when V_{tun} remains constant during the tunneling phase, as shown in 46b. The analysis is very similar to the LTP case, except for the delay between the output spike and the start of the tunnel phase, t_2. We assume that the tunneling current is determined by the gate voltage at the start of the tunneling phase. When the input occurs before the output, the gate voltage is already at the highest possible voltage resulting in the lowest possible voltage across the tunneling oxide, resulting in a very small tunneling current. When $t_d < 0$, the tunnel pulse overlaps with the pre-synaptic gate waveform, and the tunneling oxide is exposed to a larger voltage. This results in an exponentially larger tunneling current compared to the case when $t_d < 0$. For the fixed tunneling pulse, the tunneling current depends only on the voltage at the gate input at the time of tunneling (34), which can be related to t_d and the slope of the pre-synaptic waveform. Note that even without an input spike, there is a certain baseline tunneling current, which is exponentially smaller than when there is an input spike. Therefore even for large time delays between input and output spikes, there would be some tunneling that occurs during the learning algorithm.

Also, for an STDP type learning algorithm which uses a combination of injection and tunneling, injection parameters can be chosen such that its effect would dominate

in the case when the output spike occurs after the input spike. Next, we consider the case when the tunneling voltage is varied with time, to obtain a greater time dependence. The transient used on the tunneling line is shown in Fig. 46d. The tunneling voltage is allowed to slowly decay with time with a constant slope. We assume that the tunneling current is set by the maximum voltage across the tunneling oxide during the tunnel phase. When $t_d < 0$, the tunneling waveform overlaps with the pre-synaptic gate input and the tunneling current is determined by the largest voltage across the tunneling oxide which is determined by t_d and the slope of the tunneling voltage alone. For $t_d > 0$, the maximum voltage across the tunneling oxide is at the start of the tunnel phase, and the tunneling current depends on the voltage on the gate input, which in turn depends on t_d and the slope of the pre-synaptic input.

5.2.3 STDP Learning Algorithm

STDP has been found in systems ranging from the hippocampus, barrel cortex, and visual cortex [13]. Like LTP and LTD, the total scheme of how STDP works in biology has not been discovered. Thus, many descriptions of synaptic weight changes exists. Here we present an implementation of STDP which produces results similar to what has been observed in biology. We modify the timing diagram as shown in Fig. 47 to select an algorithm for the STDP learning rule. At the occurrence of a post-synaptic spike, a program phase consisting of an injection pulse followed by a tunnel input is applied. For $t_d < 0$, the input spike occurs after the output spike and hence the gate voltage remains at its highest possible voltage during the injection phase. However, the tunnel input overlaps with part of the triangular waveform at the gate. For sufficiently small negative delays, the input spike occurs during the tunneling phase of the algorithm, resulting in a large voltage across the tunneling oxide. Thus, for these values of t_d, the tunneling effect is dominant and we get a net negative change in

weight. For large negative values of t_d, there is a baseline tunneling, resulting in a net negative change in weight as discussed in the section on LTD modeling. When $t_d > 0$, the input spike occurs before the output spike, and during the injection phase the gate voltage is at its lowest point. During the tunnel phase, the gate voltage increases back to its initial value, thereby exponentially reducing the tunneling current. So for these values of t_d, the injection effect dominates resulting in a positive change in weight.

5.3 Mathematical Model

The equation for the drain current of a sub-threshold saturated pFET equation whose well is tied to V_{dd} is given by

$$I_d = I_{so} e^{\kappa(V_{dd} - V_{fg})/U_T} \tag{36}$$

where I_d is the drain current and U_T is the thermal voltage kT/q. Since the adaptation timescale is much slower than the computation timescale, we can expand (V) into its constant offset value (V_o), a fast timescale voltage change $(\triangle V)$ and a slow timescale voltage change $(\overline{V})$. As a result, we define

$$V_{fg} = V_{fgo} + \triangle V_{fg} + \overline{V_{fg}} \tag{37}$$

where V_{fg0} is the constant bias part of the floating gate voltage, analogous to a DC operating point that we expand around. We define $\triangle V_{fg}$ as the fast timescale change due to capacitive coupling; this term for a change in V_g is

$$\triangle V_{fg} = \frac{C}{C_T} \triangle V_g \tag{38}$$

where C is the capacitance between the gate and the floating node and C_T is the total capacitance at the floating gate node. We define $\overline{V_{fg}}$ as the slow timescale change in the floating-gate voltage which relates to the charge stored on the floating-gate device, and effectively defines the weight of the synaptic device as

$$w = exp(\frac{-\kappa \overline{V_{fg}}}{U_T}) \tag{39}$$

Since there is no slow change component to V_d, V_g and V_{tun}, we expand these voltages as $V_{tun} = V_{tuno} + \triangle V_{tun}$, $V_d = V_{do} + \triangle V_d$ and $V_g = V_{go} + \triangle V_g$. All of these results lead to the resulting model equation for the saturated, sub-threshold floating-gate pFET device as

$$I_d = I_{bias} w e^{-\triangle V_g / V_{gc}} \tag{40}$$

where V_{gc} is $U_T C_T / \kappa C$, which is effective voltage change required to increase the source current by an e-fold. In order to determine the update equation for the weight, which involves the slow timescale, we write

$$C_T \frac{dV_{fg}}{dt} = C_T \frac{d\overline{V}_{fg}}{dt} = I_{tun} - I_{inj} \tag{41}$$

where

$$I_{inj} = I_{inj0} (\frac{I_d}{I_{s0}})^{\alpha} e^{\triangle V_{ds} / V_{inj}} \tag{42}$$

$$I_{tun} = I_{tun0} e^{(V_{tun} - V_{fg}) / V_{ox}} \tag{43}$$

where $\alpha = 1 - U_T / V_{inj}$ and $\beta = U_T / \kappa V_{ox}$. Using (65) and (66), (64) can be re-written as

$$\begin{aligned} \frac{dw}{dt} &= A w^{1+\alpha} e^{-\alpha \triangle V_g / V_{gc}} - B w^{1+\beta} e^{-\beta \triangle V_g / V_{gc}} \\ A &= \frac{\kappa I_{inj0}}{C_T U_T} (\frac{I_{bias}}{I_{s0}})^{\alpha} e^{\triangle V_{ds} / V_{inj}} \\ B &= \frac{\kappa I_{tun0}}{C_T U_T I_{s0}} (I_{bias})^{\beta} e^{(V_{tun} - V_{dd}) / V_{ox}} \end{aligned} \tag{44}$$

For solving the resulting differential equations, we start by assuming the injection and tunneling currents do not vary significantly over the region of interest. This assumption is reasonable when solving for functions with strong exponentials, which we have in this case. Therefore, from (64) and (62), for small changes in weight,

$$\begin{aligned} \overline{V_{fg}}(t) - \overline{V_{fg}}(t=0) &= \frac{U_T}{\kappa} \log(w_o + \triangle w) - \frac{U_T}{\kappa} \log(w_o) \\ &\approx -\frac{U_T}{\kappa} \frac{\triangle w}{w_o} \\ &= \frac{T}{C_T} (I_{tun} - I_{inj}) \end{aligned} \tag{45}$$

5.3.1 LTP model

For the LTP model, we first consider the case when the output spike occurs after the input spike. When $t_d > t_1$, where t_1 is the width of the linearly decreasing portion of the gate voltage, we can write this result as

$$\triangle V_g = -s_1 t_1 + s_2(t_d - t_1) \tag{46}$$

where s_1 and s_2 are the falling and rising slopes of the pre-synaptic gate waveform. Substituting (46) in (67), we get

$$\begin{aligned} \triangle w &= A w^{1+\alpha} T_{inj} e^{(-\alpha(s_2(t_d - t_1) - s_1 t_1))/V_{gc}} \\ &= A_1 w^{1+\alpha} T_{inj} e^{-\alpha t_d/\tau_{rise}} \end{aligned} \tag{47}$$

where $A_1 = A e^{\alpha(s_1+s_2)t_1/V_{gc}}$ and T_{inj} is the width of the drain pulse. Next, we consider the case when $t_d < t_1$ resulting in a gate voltage determining the injection current as $\triangle V_g = -s_1 t_d$. We approximate the solution in this region as

$$\triangle w = A w^{1+\alpha} T_{inj} e^{\alpha s_1 t_d/V_{gc}} = A w^{1+\alpha} T_{inj} e^{\alpha t_d/\tau_{fall}} \tag{48}$$

where τ_{fall} characterizes the initial transient waveform having typical values of 0.1 - 0.2 ms. Combining the two pieces of the solution, we can write an interpolation equation for the LTP case.

$$\triangle w = \frac{w^{1+\alpha} T_{inj}}{A^{-1} e^{-\alpha t_d/\tau_{fall}} + A_1^{-1} e^{\alpha t_d/\tau_{rise}}} \tag{49}$$

5.3.2 LTD model

For LTD with the constant tunnel pulse, the analysis is very similar to the LTP case, except for the delay between the output spike and the start of the tunnel phase, t_2. We assume that the tunneling current is determined by the gate voltage at the start

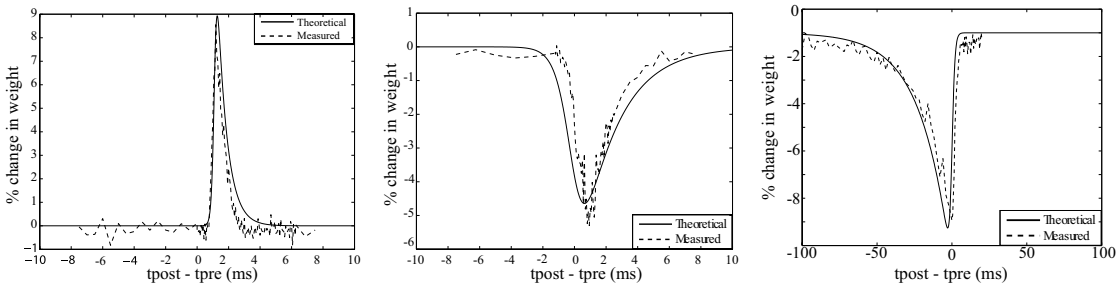


Figure 48: **Learning Experiments:** (a) Measured LTP with $V_{dd} = 4.2V$ and $T_{inj} = 25\mu s$. Note that the exponential decay of the change in weight is related to the rise time of the triangular waveform at the gate input. Predicted change in weight is plotted using (49).(b) LTD with fixed tunnel pulse, $V_{tun} = 14.9V$, $T_{tun} = 150\mu s$. The time constant of the exponential change in weight is related to the rise time of the triangular waveform at the gate input. Expected change in weight using LTD model in (52). (c) LTD with linearly decreasing tunnel voltage, with slope $17.5V/s$, starting at $V_{tun} = 14V$. Predicted weight from (54), time dependence of the LTD model is determined by the rate of decrease of the tunnel voltage and the rise time of the triangular gate input.

of the tunneling phase. For the output spike occurring before the input spike ($t_d < 0$) and $-t_d + t_1 < t_2$, we can write $\triangle V_g = -s_1 t_1 + s_2(t_2 - \mid t_d \mid -t_1)$.

$$\begin{aligned} \triangle w &= -Bw^{1+\beta} T_{tun} e^{-\beta(s_2 t_d - (s_1+s_2)t_1)/V_{gc}} \\ &= -B_1 w^{1+\beta} T_{tun} e^{-\beta t_d/\tau_{rise}} \end{aligned} \tag{50}$$

where $B_1 = Be^{-\beta((s_1+s_2)t_1+s_2 t_2)/V_{gc}}$. When $t_d + t_2 < t_1$, the tunnel pulse starts during the decreasing part of the gate waveform. We can write $\triangle V_g = -s_1(t_d + t_2)$ resulting in

$$\triangle w = -B_2 w^{1+\beta} T_{tun} e^{\beta t_d/\tau_{fall}} \tag{51}$$

where $B_2 = Be^{\beta s_1 t_2/V_{gc}}$. An interpolated expression can be written for LTD as

$$\triangle w = -\frac{T_{tun} w^{1+\beta}}{B_1^{-1} e^{\beta t_d/\tau_{rise}} + B_2^{-1} e^{-\beta t_d/\tau_{fall}}} \tag{52}$$

When the tunneling voltage is varied with time, we obtain for $t_d + t_2 < 0$,

$$\triangle w = -B_3 T_{tun} w^{1+\beta} e^{t_d/\tau_{cp}} \tag{53}$$

$B_3 = e^{-s_3(t_1+t_2)/V_{ox}}$. For $t_d > 0$, the largest voltage across the tunneling oxide is set by the gate voltage, since the tunneling voltage is at its maximum at the start of the tunneling phase. Substituting $\triangle V_g = -s_1 t_1 + s_2(t_d + t_2 - t_1)$ in (34), we get $\triangle w = -B_4 T_{tun} w^{1+\beta} e^{-\beta t_d/\tau_{rise}}$, $B_4 = e^{\beta(-t_1(s_1+s_2)+s_2 t_2)/V_{gc}}$. Thus, we can write an interpolated equation for the change in weight as

$$\triangle w = -\frac{T_{tun} w^{1+\beta}}{B_3^{-1} e^{-t_d/\tau_{cp}} + B_4^{-1} e^{\beta t_d/\tau_{rise}}} \tag{54}$$

5.3.3 STDP model

We obtain the expression for an STDP model with a fixed tunnel pulse by combining (49) and (52) as

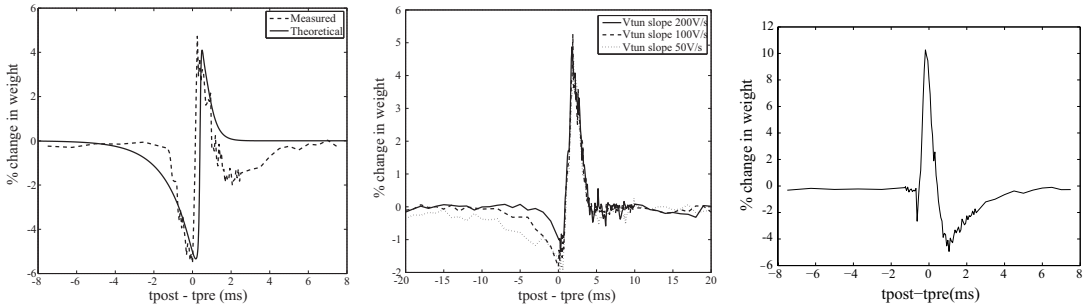
$$\begin{aligned} \triangle w = & \frac{w^{1+\alpha} T_{inj}}{A e^{-\alpha t_d/\tau_{fall}} + A_1^{-1} e^{\alpha t_d/\tau_{rise}}} \\ & - \frac{T_{tun} w^{1+\beta}}{B_1^{-1} e^{\beta t_d/\tau_{rise}} + B_2^{-1} e^{-\beta t_d/\tau_{fall}}} \end{aligned} \tag{55}$$

For the STDP algorithm shown in Fig. 47, we add (49) and (54) to get

$$\begin{aligned} \triangle w = & \frac{w^{1+\alpha} T_{inj}}{A e^{-\alpha t_d/\tau_{fall}} + A_1^{-1} e^{\alpha t_d/\tau_{rise}}} \\ & - \frac{T_{tun} w^{1+\beta}}{B_3^{-1} e^{-t_d/\tau_{cp}} + B_4^{-1} e^{\beta t_d/\tau_{rise}}} \end{aligned} \tag{56}$$

5.4 Measurements from Spike Based Learning Experiments

Fig. 48a shows the change in synaptic weight for different delays between pre- and post-synaptic spikes. The programming algorithm followed is shown in Fig. 46a. When an output spike occurs, the chip is turned into "PROGRAM" mode and injected for a short duration. The synapse device is operated at a high voltage ($V_{dd} = 4V$) in order to eliminate ramp-up delays, and injection is achieved by pulsing down the drain terminal. When an input spike occurs, the pre-synaptic computation block generates a triangular waveform that results in an EPSP lasting 1 ms, similar to biological timescales. For $t_d < 0$, there is no current through the device, hence no



(a) (b) (c)

Figure 49: **Learning Experiments:** (a) STDP: Using a constant tunneling pulse with $V_{tun} = 15V, V_{inj} = 4.2V, T_{tun} = 200ms, T_{inj} = 100\mu s$. (b) STDP: Using a linearly decreasing tunnel voltage. Data presented here is from an experiment with $T_{tun} = 200ms$, starting at $V_{tun} = 14V$ with 3 different slopes. Note that the slope is inversely proportional to the duration of the LTD effect. (c) Modified algorithm to obtain a reversed STDP learning rule, with $V_{tun} = 14V, V_{inj} = 4.2V, T_{tun} = 2.5ms, T_{inj} = 50\mu s$.

injection occurs. For positive values of t_d, V_g is sufficiently low during injection to increase the current through the device exponentially and there is an increase in the synaptic weight. For $t_d \gg 0$, the gate voltage has risen back up, thereby reducing the drain current and the change in weight. Fig. 48a also shows the predicted change in weight, obtained using (49). Fig. 48b shows the change in synaptic weight for an LTD-type learning rule. At the occurrence of an output spike, the synapse is tunneled for a short duration. Tunneling occurs when the barrier across the tunneled oxide is low enough for electrons to tunnel through. Even for $t_d \geq 0$, we can observe a decrease in synaptic weight since the tunneling pulse overlaps the triangle gate waveform. The largest change in weight occurs when the tunnel pulse overlaps the minimum of the gate voltage. Fig. 48b shows the measured change in weight for different delays between input and output with the LTD algorithm shown in Fig. 46b. The tunneling voltage used here is a constant pulse of fixed width. Fig. 48b also shows predicted change in weight obtained using (52). For the tunnel control block shown in Fig. 45, we would have a small charge pump for the tunneling device, such as a Dickson charge pump. It has been previously experimentally demonstrated

in [42] that a four stage charge pump is sufficient for generating the tunneling voltage. Given that the tunneling current is very small, we do not expect loading of the charge pump to be an issue. Due to natural current source leakage from the "OFF" charge pumping elements, charge will slowly leak off the tunneling junction, giving us the desired slow movement of tunneling voltage with time. Alternatively, an external high voltage power supply can be switched in or out depending on the timing of the LTD algorithm. In these experiments we use an external power supply. Fig. 48c shows the change in synaptic weight for the learning algorithm described in Fig. 46d. With a tunneling voltage decreasing with a slope of $17.5V/s$, we see that the LTD effect extends for a much longer duration as compared to the algorithm in Fig. 46b. We also see the predicted change in weight using (54) matches the measured data well.

5.4.1 STDP Learning Experiments

Next, the change in weight for different delays between input and output is plotted for an STDP algorithm shown in Fig. 49a. The expected change in weight from (55) is also plotted. When an output spike occurs, the drain of the synapse device is pulsed for a duration of $T_{inj} = 100\mu s$, then the tunneling line is pulsed for $T_{tun} = 2ms$. The results presented here are for an experiment with $V_{dd} = 4.2$V and $V_{tun} = 15$V.

For $t_d > 0$, the injection effect dominates, since the gate voltage is at its lowest point during the injection phase. Also, the gate voltage starts increasing during the tunneling phase and parameters can be chosen such that injection overrides the tunneling currents. Hence we expect a positive change in weight for these delays. The negative change in weight we see for some positive delays in Fig. 49a is due to a stronger tunneling phase than desired. The parameters for injection and tunneling were chosen to cause $< 10\%$ change in weight after one learning event. For a constant tunnel pulse, the time dependence of the LTP and LTD portions of the learning rule are roughly equal. When a linearly decreasing tunneling voltage is used during the

tunnel phase of the programming algorithm, we expect a longer time dependence for the LTD, which is similar to learning rules seen in the hippocampus. By varying the slope of the tunneling voltage, the timing dependence during LTD can be changed. Fig. 49b shows the change in weight for different delays between input and output spikes while using the programming algorithm shown in Fig. 47.

Fig. 49c shows the learning rule obtained when the programming algorithm consists of a tunnel phase followed by an injection phase. For $t_d < 0$, the gate voltage is at its highest level during the tunnel phase, thus injection dominates. For positive delays, the tunnel phase overlaps with the lowest gate voltage, resulting in a large decrease in weight. The programming algorithm described in this research provides a general framework for implementing several different learning rules.

5.5 Conclusion

We have presented results from a silicon synapse capable of implementing STDP learning. The small size of the synapse and supporting circuitry that can be shared across several synapses make this an attractive implementation and allows a high level of integration. It is possible to achieve a wide set of biological learning rules by modifying the programming algorithm for the synapse. [4, 44, 48, 60, 63, 78] have demonstrated learning synapses, but the one presented in this work has the added advantage of density over other implementations. In an area of 1.2mm x 1.7mm, we have 20,000 synapses. This is an important consideration in realistic network simulations, since synapses outnumber neurons 1000 to 1 in biology [18]. The learning rule implemented here is also a weight-dependent STDP rule as opposed to bimodal, and a function of difference in pre- and post-synaptic spike times as opposed to a rate-based learning rule proposed in [60]. Since our synapse structure is based on a floating-gate device, small changes in weight can be stored in a non-volatile manner. Thus, our learning rule is closer to the one observed in [9], as opposed to bimodal

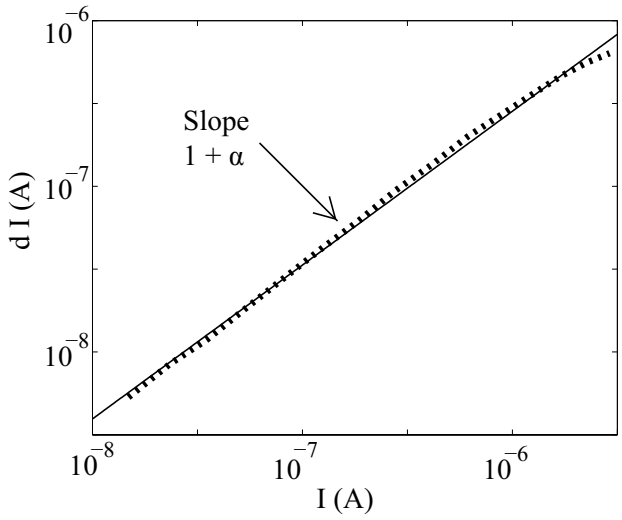


Figure 50: **Injection Characterization:** Plot of change in drain current versus initial current (dashed line) after an injection pulse with parameters $V_{inj} = 4.2V$ and $T_{inj} = 1ms$. A linear fit (solid line) to the data (in log scale) suggests a power law dependence on initial weight for the STDP learning rule.

STDP implementations in [4, 48]. We have derived a mathematical model for the learning rule implemented in the synapses, which gives a good fit to measured results. This model can be integrated with a simulation tool to study properties of networks with this type of learning.

Another distinguishing feature of the proposed structure is that it results in an STDP model that has an inherent weight dependence. While there is a lack of consensus in the neuroscience community about the weight dependence of the STDP model, we hypothesize using the model derived from the STLS, that the change in weight has a power law relation to the initial weight. This is seen from experimental data for the characterization of the injection process shown in Fig. 50. The plot shows the change in the measured drain current versus initial drain current for a device injected with a $V_{inj} = 4.2V, T_{inj} = 1ms$. The plot is linear in the log scale and hence indicates a power law relationship, $\triangle w \propto w^{1+\alpha}$ which would be useful to compare biological data.

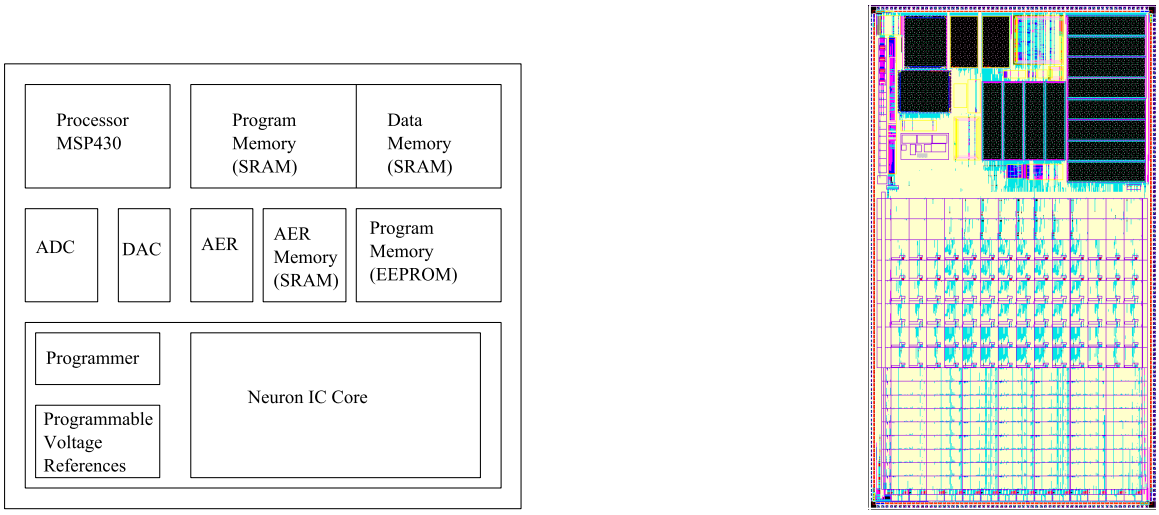
CHAPTER VI

SYSTEM IMPLEMENTATION

A classifier system involving all three components described previously would require a developing a board with a RASP 2.9 IC and a Neuron2 IC. A second option however, is to integrate all elements required to build these classifiers on a single IC. The following chapter describes my work towards building such an IC. The advantages of an integrated implementation is immediately apparent in terms of power and area.

6.1 RASP 3.0N

The chip titled RASP 3.0N, integrates several elements present on the board on to the die including the processor, memory and peripherals. A block diagram of the RASP 3.0N and the layout of the IC submitted for fabrication is shown in Fig. 51.



(a) (b)

Figure 51: **RASP 3.0N IC:** A block diagram of the IC submitted for fabrication and the completed layout. The IC integrates a processor, memory and several peripherals with the NEURON RASP core.

A 16-bit microcontroller core (openMSP430) available in the public domain (in

Verilog) was synthesized. The processor is compatible with the Texas Instruments MSP430 microcontroller family and can execute code generated by the MSP430 toolchain. The system also integrates 16KB SRAM for program memory and 16KB for data memory. A 16-channel DAC is also integrated along with the core, a few channels being reserved for the drain and gate voltages during programming, and the rest available for setting inputs and biases. 2 high-speed ADCs are also integrated with the core, allowing possibilities of using the MSP430 as a co-processor along with the RASP 3.0N. A high level block diagram of the RASP 3.0N is shown in Fig. 52.

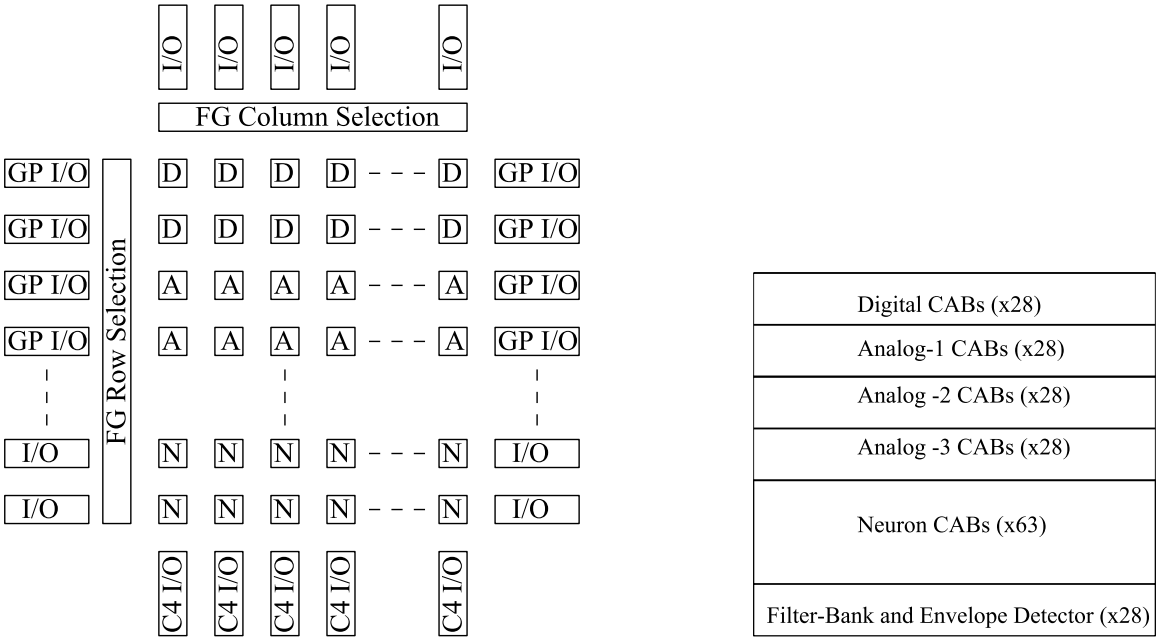


Figure 52: **CABs in the Neuron RASP IC Core:** A combination of digital, analog and neuron CABs are present in the RASP 3.0N core. Together with a filter-bank front-end whose outputs are directly routed into the interconnect, all blocks for building the classifier are integrated on a single IC.

The idea of integrating the neurons with the analog and digital computing elements in the RASP IC is two-fold. Firstly, it allows seamless integration of all elements in the classifier. Secondly, it opens the possibilities of using the neurons as computing elements in signal processing applications. Thirdly, the architecture of the RASP 3.0N core is supported by Versatile Place and Route (VPR) tool, is a

standardized tool for FPGA routing and will allow quick and optimal placement of components.

6.1.1 Neuron RASP Core

The RASP 3.0N core has 28 Digital tiles, 84 Analog tiles (each having specialized blocks), and 63 Neuron tiles. The basic tile is depicted in Fig. 53. Each tile consists of the global interconnect and the CABs. The global interconnect consists of the C-Block that makes connections from CABs to the interconnect and the S-Block (switch block) that is used for routing. Within the CAB, there exists the local interconnect, which allows all-to-all connectivity between the components. The array was designed such that the analog and digital CABs have 24 I/O. This choice reflects a tradeoff between the number of I/O and the size of the local interconnect within each block. An increased local interconnect also increases routing parasitics. Each Digital CAB consists of 8 BLEs and local interconnect. The BLE itself comprises of a Look-up Table (LUT) and a flip-flop whose inputs and clocks are routable.

The analog and digital tiles have general purpose I/O blocks terminating the tiles, which allow analog or digital signals in/out of the tiles. The I/O blocks terminating the neuron tiles are basic I/O which connect the global interconnect to the AER in/out blocks. At the bottom of the array is the $C4$ I/O block, which consists of programmable filterbanks. The $C4$ I/O block is reconfigurable, since its inputs may be from an external microphone or from the array itself. The outputs from the filterbank can be routed into the array for further processing, or be routed out to pads.

The filterbank is the first stage of the auditory processing. The $C4$ I/O block has two outputs - a filterbank output and an envelope detector output. These outputs may be processed further to do sub-banded speech enhancement using the blocks in the CABs shown in Fig. 54. The blocks in Analog CAB2 and CAB3 can be used for

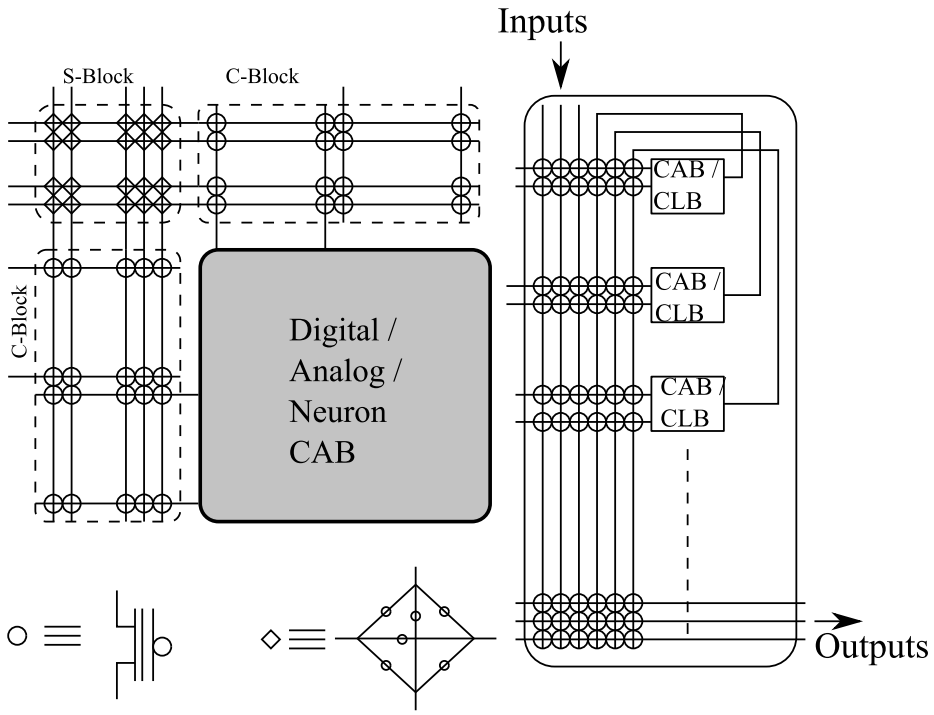


Figure 53: **Tiles in the RASP 3.0N IC:** The tiles consist of CABs and the global interconnect. Within each tile, a local-inteconnect or switch matrix is used to make connections between blocks.

"speech quality estimation" and "gain control", as discussed in Chap. 2. The Analog CAB3 has been designed to include elements required for building VMM+WTA classifiers and directly interfacing with the neuron elements, without involving the AER blocks resulting in a low-latency and low-power approach. The blocks in CAB3 include $4-input$,$4-output$ WTA elements, and discrete pFETs that can be used to convert the WTA outputs into digital "events". The event to synaptic input conversion block is similar to the gate waveform shaper block in the neuron itself, used to make connectivity between neurons. The outputs from the WTA can be directly connected to the synaptic input generator blocks within CAB3, and the resulting output from CAB3 can serve as synaptic inputs to the neurons.

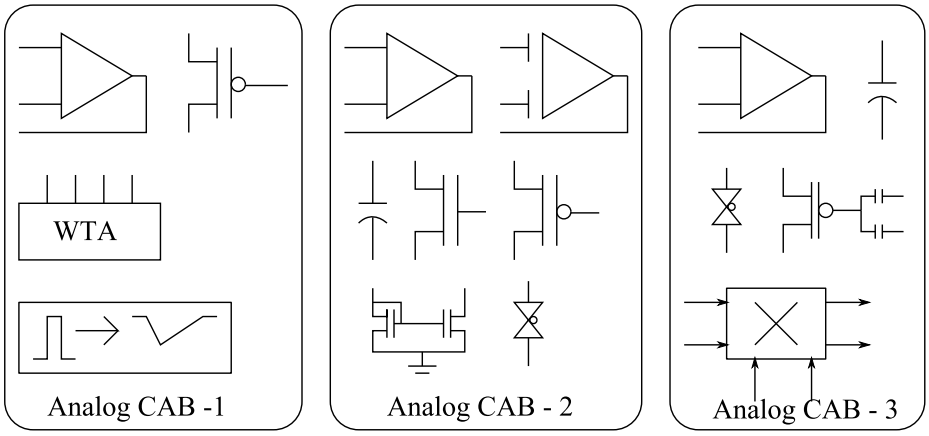


Figure 54: **Analog CABs:** Three types of analog CABS are present in the RASP 3.0N IC, with specialized blocks for building VMM+WTA classifiers and signal-conditioning blocks converting their outputs to synaptic inputs to the neurons.

6.1.2 Neuron Tile

Since the RASP 3.0N architecture supported only 24 inputs and we require more inputs into each neuron, the neuron CAB was made twice as wide as the analog/digital CABs, thereby increasing the number of inputs by 50%. The number of inputs doubled in the dimension that also doubled, while it stayed the same in the other dimension. To further increase the number of inputs, the number of global routing tracks would have to be increased to ensure that all inputs can be routed simultaneously. Another approach to increase the number of inputs is to "hardcode" certain inputs from neighbor and nearest-neighbor neurons, leaving the programmable inputs for neurons farther away. Examples of specific applications and the number of inputs are needed to make a decision on whether this architecture provides a favorable trade-off between ease of routing and number of synaptic inputs. The synaptic inputs and the dendrites which make up the equivalent "local interconnect" in the neuron CAB is shown in Fig. 55.

The architecture chosen for the dendrites differs from that chosen in the Neuron2 IC described previously in Chap. 4. The synaptic inputs in the Neuron2 IC were present only in the periphery of dendrites, reducing the number of synaptic inputs

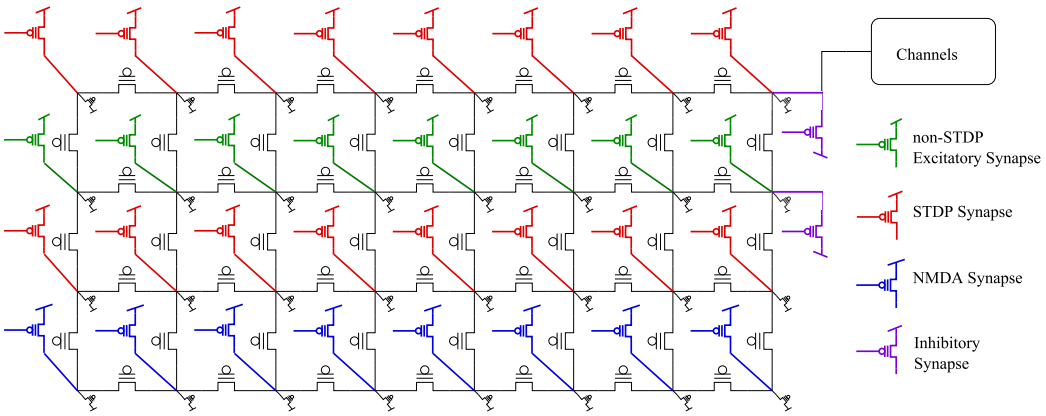


Figure 55: The Neuron CAB: synapses are distributed throughout the dendrites to increase the density of synapses per unit dendrite area.

available per unit area of the dendrites. Further, the dendritic classifier also requires a synaptic input per dendritic compartment, which was not possible with the Neuron2 IC. Hence, the structure of the dendrite has been modified as shown in Fig. 55.

The neurons have 34 inputs and 2 outputs : 16 stdp inputs, 8 nmda inputs, 8 non-stdp inputs, 2 inhibitory inputs. The outputs are soma output and intermediate dendrite node output. Several improvements have been made in the RASP 3.0N IC, compared to the Neuron2 IC. Many of these are related to implementing learning synapses. In the Neuron2 IC, even during normal operation, the learning synapses were on a separate injection-level supply. During learning events, the drains of the synapses were pulsed to a lower voltage to allow injection.

There were several disadvantages to such an implementation. Firstly, the synapses sources and the gates (inputs to the synapses from the global interconnect) had to be on injection level supplies during normal circuit operation. Besides complicating the design and adding protection diodes to ensure no unintended injection occurred, it also affected the column selection and row selection circuitry. Further, a high voltage supply caused an increased power draw due to leakage currents. If the injection voltage supply is implemented using a switching converter, that also contributes to noise during normal circuit operation. The design would be complicated even further

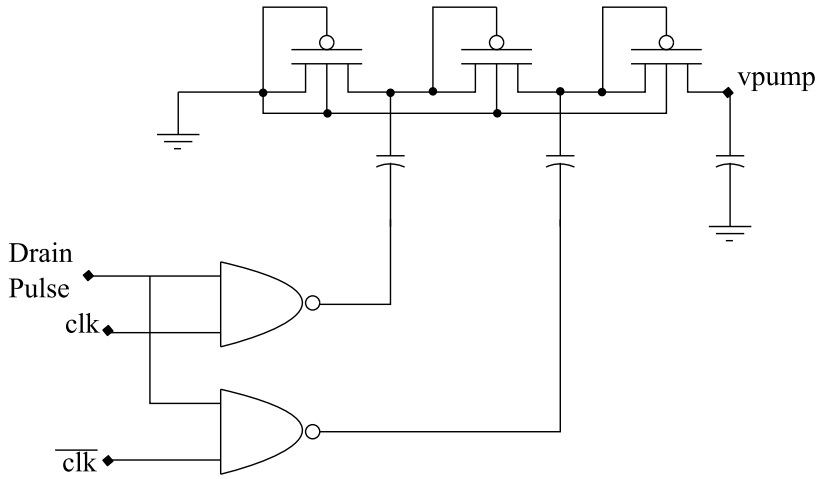


Figure 56: **Charge Pump:** An integrated negative chargepump circuit eliminates the need for noisy high-voltage supplies that allow run-time injection for synaptic learning.

while trying to interface with analog and digital tiles which do not require injection level supplies.

To avoid these drawbacks and to simplify the design process, the Neuron CABs in the RASP 3.0N IC were designed to operate at the same clean $2.5V$ supply as the rest of the elements. Additionally, to support learning in the synapses, local negative-chargepumps were included in each neuron which will provide a sufficient field across the drain and source terminals to cause injection. The circuit diagram of the negative chargepump is shown in Fig. 56.

The microprocessor clock can be used as the clock for the chargepump. When STDP learning is enabled and the drain pulse is active for a particular neuron, the clocks are gated to the chargepump, allowing a negative voltage to be built up at V_{pump}. The clocks to the chargepump can be muxed so as to be generated from the array itself. The digital blocks in the array can be used to build a non-overlapping clock generator, and then routed to the chargepump, giving more flexibility. Simulations showed a pumping action of $3V$ within $1ms$ of the start of the pumping action. However, a load current (from the synapses) reduced the pumped voltage. This is expected to be an issue as the number of active synapses increases, but this may also

model a built-in regulatory mechanism where neurons with several active synapses or synapses with large weights experience a lower degree of potentiation than those neurons with fewer synaptic inputs.

NMDA receptors in synapses are thought to play an important role in synaptic plasticity. A sufficient pre-synaptic excitation causes NMDA receptors to be activated and increase synaptic efficacy. This effect is modeled by a new type of synapse implemented in the RASP 3.0N IC, which we call the "NMDA Synapse". Studies on dendritic trees have also revealed that NMDA synapses are a key component for obtaining robust directional selectivity. These synapses have an inbuilt positive feedback mechanism that model higher synaptic strength for those synapses that have a high local dendritic potential. The circuit diagram of the NMDA synapse is shown in Fig. 57. An common-source amplifier whose gain is set by the ratio of the input capacitance of the floating gate to the overlap capacitance is used to amplify the local dendritic potential before feeding it back to the second control gate on the synapse. The inverting characteristic of the common-source amplifier is desirable, since the synapse is implemented using a pFET device.

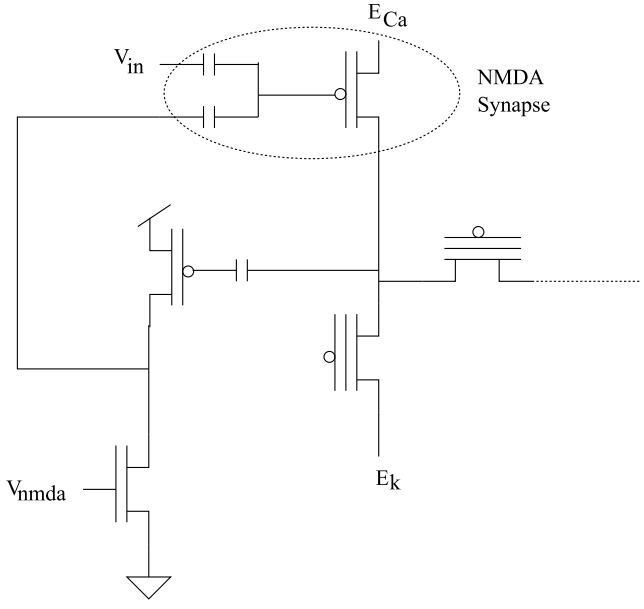


Figure 57: **NMDA synapse:** Circuit model for synapses with NMDA receptors.

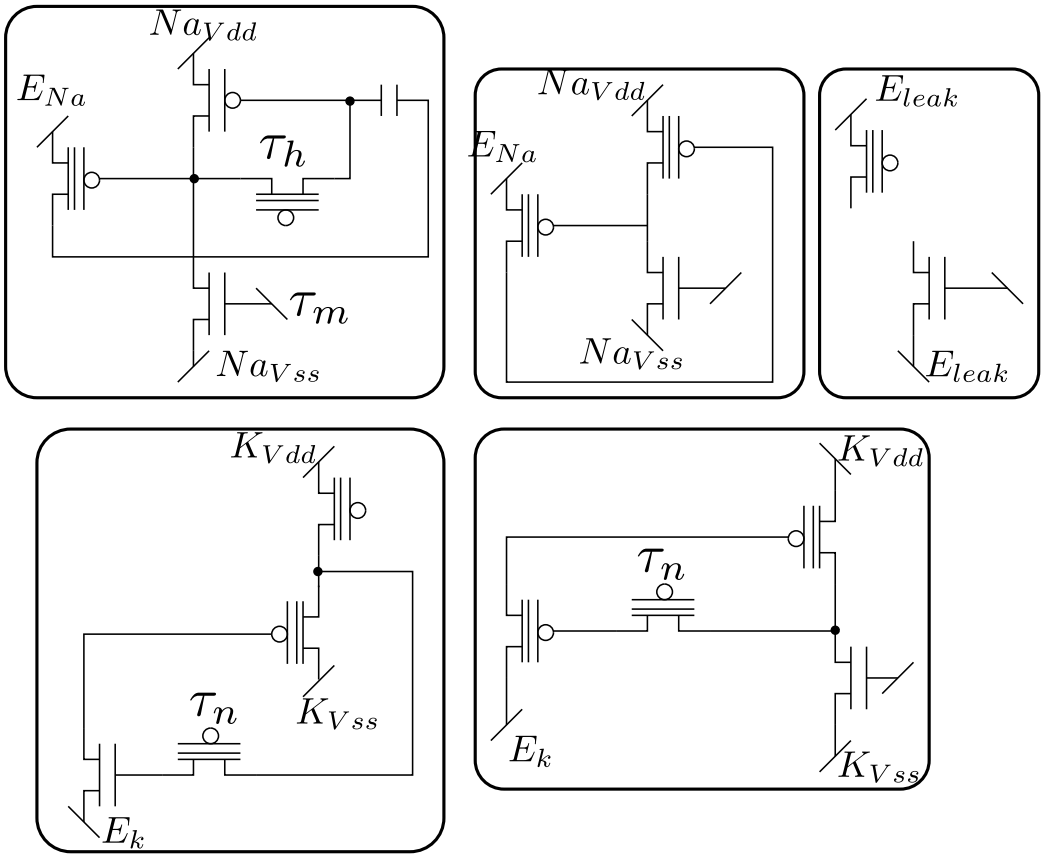


Figure 58: **Channels in the Soma:** The RASP 3.0N IC includes channels that exhibit a Hopf bifurcation and a Saddle-node bifurcation.

The other component of the Neuron CAB is the soma, which consists of programmable channels and the WTA block. In addition to the Hopf channels which were included in the Neuron2 IC, extra channels that show saddle-node dynamics are also included in the soma. Effectively, the soma has 2 Sodium channels, 2 Potassium channels and 2 leak channels with programmable parameters. The circuit schematics of the channels in the RASP 3.0N is shown in Fig. 58. Besides the channels, the WTA block is also included in the soma, with two key modifications from the Neuron2 IC. The source of the input device to the WTA has been modified to be at E_k and the output from the WTA can be converted into digital events. The Neuron output is selectable between the output from the membrane and the WTA using a floating-gate memory device.

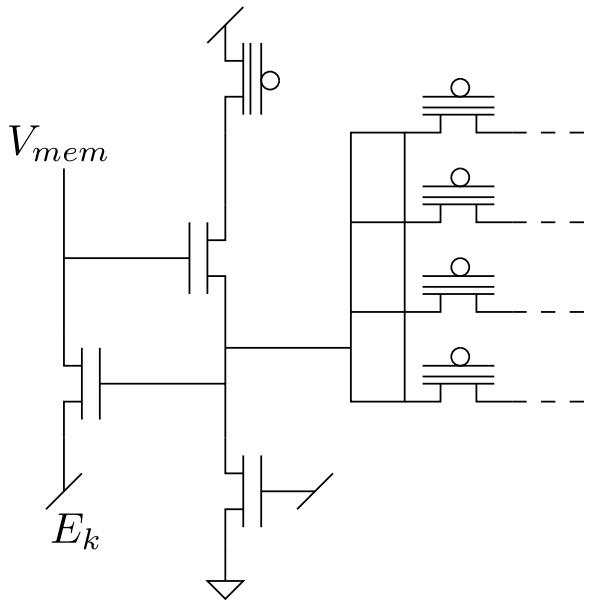


Figure 59: **Modified WTA block:** The WTA block has been modified from the Neuron2 IC, to allow digital outputs.

The choice of integration platform being the standard FPGA architecture with a Manhattan-style global and local interconnect allows us to leverage several existing tools from the suite of tools for automatic place and route for FPGAs. VTR, an open-source academic software takes inputs in verilog, and given the FPGA architectural description, synthesizes the logic, places and routes the elements. The analog and neuron elements can be treated as black boxes, and taking intermediate outputs from the VTR tool flow, another tool named VPR2P can be used to place and route these blocks. This streamlines the tools for automatic placement and routing, given cost functions of parasitic capacitance/area etc, and makes use of existing tools instead of requiring costly tool development.

CHAPTER VII

CONCLUSION

The previous chapters detail my work that leads up to the system implementation of neuromorphic classifier architectures. The goal of this dissertation is to design hardware that enables analog and neuromorphic computing. Digital computing efficiencies observed in commercial ICs have reached an efficiency wall due to device mismatch, whose effects are more prominent in small feature sizes. There is a great need for alternate computing strategies that allow us to get past the efficiency wall. Analog processing has been shown to be more efficient than digital, particularly for applications such as multiplication, filtering, FFT etc. However, custom analog has fixed functionality and is too expensive to design and fabricate. We take a programmable analog approach, which gives flexibility in functionality and allows mismatch compensation. Further, we explore neuromorphic strategies by building silicon models of computational primitives in the brain - neurons, synapses and dendrites. Neuromorphic computing has to be more efficient for engineering applications for it to be a viable alternative to digital computing. As a result, we take a low-power approach to the design of the neuromorphic ICs by taking inspiration from FPGAs.

In an effort to target a speech recognition application using programmable analog and neuromorphic classifiers, I first described an auditory front-end that generates sub-banded enhanced analog outputs. A feature/symbol extraction block can be implemented using a VMM and WTA structure. A phoneme or word recognizer can be built using the dendrites available in the Neuron2 chip, as described in [24].

In Chapter 2, an auditory front-end implementation consisting of non-linear filtering blocks was described. Sub-banded processing blocks with expansive non-linearity

or automatic gain control based on signal activity can result in noise suppression. The system framework described also supports other applications such as speech activity detection, hearing aids and classifier front-ends. An extension of this work would be to test the speech enhancement front-end with a software HMM classifier implementation to observe lower error percentages.

In Chapter 3, a powerful, compact, programmable classifier was described. This classifier was implemented with routing elements on the FPAA, and used a WTA as the decision-making element. It has been shown that this classifier is capabale to implementing linear and nonlinear decision boundaries. An important contribution of this work is that we have halved the computing resources required for implementing non-linear classifiers, as compared to a NN implementation.

In Chapter 4, a neuromorphic IC with a low-power scalable architecture that allows investigation of neural and dendritic computation is described. This platform is designed to implement a dendritic wordspotter network and demonstrates properties of dendrites that are critical for implementing the wordspotter, namely directional selectivity and spatio-temporal summation.

An algorithm for the implementation of the STDP learning rule observed in synapses is described in Chapter 5. This algorithm has been successfully implemented in the Neuron2 and Neuron1 ICs.

Finally, Chapter 6 describes a system implementation consisting of an IC that combines the auditory front-end, feature extraction and dendritic classifier blocks. During the course of this work, I have also been involved in several other projects which have not been described in this document. My important contributions and the list of collaborators have been listed below.

7.1 List of Contributions

- Implementation of auditory front-end for speech recognizers on the RASP 2.8a.

- Development of learning algorithm to implement biological learning rules in CMOS floating gate synapses.
- Implementation of VMM+WTA classifiers on the RASP 2.9v.
- Design and layout of RASP 3.0N IC. Collaborators: All ICELAB members.
- Design and layout of a Neuron2 chip, consisting of biological channels, programmable dendrites, active channels, learning synapses and AER infrastructure. Collaborator: Richard Wunderlich.
- Design, layout and infrastructure development of the general FPAA system. Collaborators: Arindam Basu, Stephen Brink, Craig Schlottmann, Scott Koziol, Csaba Petre and Christopher Twigg.
- Design and layout of the Rasp 2.8b chip, titled the BioFPAA. Collaborator: Arindam Basu.
- Design and layout of the Neuron1 chip. Collaborators: Stephen Brink, Richard Wunderlich, and Arindam Basu.
- Contributed to design of Analog memory peripheral. Collaborators: Farhan Adil, Suma George, and Richard Wunderlich.

APPENDIX A

A COMPACT VMM

Consider a four quadrant VMM cell, shown in Fig. 24a. We start with the signed input x and the desired multiplication $y = w * x$, where w is a signed weight. The core of the VMM is a current multiplication with the input current being expressed as $I_{in} \propto x$. In our multiplier structure, currents are unidirectional but we desire four quadrant behavior. This is achieved by using differential input currents. The signed input x is encoded as

$$\begin{aligned} I_{in,p} &= I_{in,bias}(1 + (x/2)) \\ I_{in,n} &= I_{in,bias}(1 - (x/2)) \end{aligned} \tag{57}$$

The common mode input current is given by

$$I_{in,bias} = (I_{in,p} + I_{in,n})/2 \tag{58}$$

and the signal input is

$$I_{in} = (I_{in,p} - I_{in,n}) = I_{in,bias}(x) \tag{59}$$

The output of the trans-impedance stage implementing the I-V stage can be calculated by writing the sub-threshold current equation for the transistor in feedback. We assume that the transistor bulk is tied to the power supply.

$$\begin{aligned} I_{pfet} &= I_o e^{\kappa(V_{DD}-V_{FG})/U_T} e^{-(V_{DD}-V_S)/U_T} \\ &= I_o e^{V_{DD}(\kappa-1)/U_T} e^{-\kappa V_{FG}/U_T} e^{V_S/U_T} \\ &= I_{in,bias} w e^{V_S/U_T} \end{aligned} \tag{60}$$

The constraint on the input range can be seen from (60). x is a dimensionless input and (1+ (x/2)) expresses the ratio of the input current to the bias current. Since the voltage input is applied to the source and due to the exponential dependence of the drain current on source voltage, linearization only holds for a small voltage range.

Using (60), the output of the trans-impedance stage that sets the source voltage of the input device which has a weight $w = 1$, we get

$$v_{s,p} = U_T \ln \frac{I_{in,p}}{I_{in,bias}} \quad v_{s,n} = U_T \ln \frac{I_{in,n}}{I_{in,bias}} \tag{61}$$

$$\begin{aligned} v_{s,p} - v_{s,n} &= U_T \ln \frac{I_{in,p}}{I_{in,n}} \\ &= U_T \ln \frac{1 + (x/2)}{1 - (x/2)} \end{aligned} \tag{62}$$

For small values of x, i.e. $-1 \leq x \leq 1$, $\ln(\frac{1+x/2}{1-x/2}) \approx x$ and hence, $v_{s,p} - v_{s,n} \approx U_T x$.

To generate the current inputs to the VMM, $v_{in,p}, v_{in,n}$ are applied to the negative terminal of an OTA with bias current $I_{otabias}$, used here as a V-I block. To allow values for $-1 \leq x \leq 1$, we require $I_{otabias} \geq 2I_{in,bias}$. As a result, the input currents are

$$\begin{aligned} I_{in,p} &= I_{otabias} \tanh(\kappa_{eff}(v_{in,p} - v_{ref})/2U_T) \\ I_{in,n} &= I_{otabias} \tanh(\kappa_{eff}(v_{in,n} - v_{ref})/2U_T) \end{aligned} \tag{63}$$

By using small inputs or a highly linear input stage that has capacitive dividers at the inputs, we can make a linear approximation of (63).

$$\begin{aligned} I_{in,p} &= \kappa_{eff} I_{otabias}(v_{in,p} - v_{ref})/2U_T \\ I_{in,n} &= \kappa_{eff} I_{otabias}(v_{in,n} - v_{ref})/2U_T \end{aligned} \tag{64}$$

The differential voltage input can be expressed as

$$v_{in,p} - v_{in,n} = (\frac{2U_T}{\kappa_{eff}}) \frac{I_{in,p} - I_{in,n}}{I_{otabias}} \tag{65}$$

By choosing $I_{otabias} = 2I_{in,bias}$, we obtain the relation between voltage inputs to the two VMM topologies as a function of the input x.

$$v_{in,p} - v_{in,n} = \frac{U_T}{\kappa_{eff}} x = \frac{v_{s,p} - v_{s,n}}{\kappa_{eff}} \tag{66}$$

κ_{eff} denotes the effective coupling from the OTA input to the channel of the differential pair transistors and includes any linearizing factor applied to the OTA to obtain a wide linear input range. The output current can be calculated using the pFET subthreshold equation (60) and (59) as

$$\begin{aligned} I_{out} &= I_{in,bias}(w + \Delta w)e^{v_{sp}/U_T} + I_{in,bias}(w - \Delta w)e^{v_{sn}/U_T} \\ &= (w + \Delta w) * I_{in,p} + (w - \Delta w) * I_{in,n} \\ &= 2I_{in,bias}w + I_{in,bias}x\Delta w \end{aligned} \tag{67}$$

The first and second terms in (67) represent the bias and the four quadrant multiplication terms respectively, since x and Δw can be signed.

REFERENCES

[1] Abbott, L. F. and Nelson, S. B., "Synaptic plasticity: taming the beast," *Nat. Neurosci. 3*, pp. 1178–1183, 2000.

[2] Ananthanarayanan, R. and Modha, D., "Anatomy of a cortical simulator," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pp. 1–12, ACM, 2007.

[3] Anderson, D., Hasler, P., Ellis, R., Yoo, H., Graham, D., and Hans, M., "A low-power system for audio noise suppression: a cooperative analog-digital signal processing approach," in *Proc. IEEE 10th and the 2nd Signal Processing Education Workshop Digital Signal Processing Workshop*, pp. 327–332, 2002.

[4] Arthur, J. and Boahen, K., "Learning in silicon: Timing is everything," in *Advances in Neural Information Processing Systems 18* (Y. Weiss, B. S. and Platt, J., eds.), (Cambridge, MA), MIT Press, 2006.

[5] Baskaya, F., Reddy, S., Lim, S. K., and Anderson, D. V., "Placement for large-scale floating-gate field-programable analog arrays," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 8, pp. 906–910, 2006.

[6] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C. M., and Hasler, P., "A floating-gate-based field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, 2010.

[7] Basu, A., Ramakrishnan, S., and Hasler, P., "Neural dynamics in reconfigurable silicon," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 1943–1946, IEEE, 2010.

[8] Basu, A., Ramakrishnan, S., Petre, C., Koziol, S., Brink, S., and Hasler, P., "Neural dynamics in reconfigurable silicon," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 4, no. 5, pp. 311–319, 2010.

[9] Bi, G.-Q. and Poo, M., "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength and post-synaptic cell type," *J. Neuroscience*, vol. 18, pp. 10464–72, 1998.

[10] Bliss, T. and Lomo, T., "Long-lasting potentiation of synaptic transmission in the dentate gyrus area of the anaesthetized rabbit following stumulation of the perforant path," *Journal of Physiology*, vol. 232, pp. 331–356, 1973.

[11] Borgstrom, T., Ismail, M., and Bibyk, S., "Programmable current-mode neural network for implementation in analogue MOS VLSI," in *Circuits, Devices and Systems, IEE Proceedings G*, vol. 137, pp. 175–178, IET, 1990.

[12] Brink, S., Koziol, S., Ramakrishnan, S., and Hasler, P., "A biophysically based dendrite model using programmable floating-gate devices," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 432–435, IEEE.

[13] Caporale, N. and Dan, Y., "Spike timing-dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci*, vol. 31, pp. 25–46, 2008.

[14] Cauwenberghs, G. and Pedroni, V., "A low-power CMOS analog vector quantizer," *Solid-State Circuits, IEEE Journal of*, vol. 32, no. 8, pp. 1278–1283, 1997.

[15] Chabries, D. M., Anderson, D. V., Stockham, T. G., J., and Christiansen, R. W., "Application of a human auditory model to loudness perception and hearing compensation," in *Proc. Int Acoustics, Speech, and Signal Processing ICASSP-95. Conf*, vol. 5, pp. 3527–3530, 1995.

[16] Chakrabartty, S. and Cauwenberghs, G., "Sub-microwatt analog VLSI trainable pattern classifier," *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 5, pp. 1169–1179, 2007.

[17] Chawla, R., Bandyopadhyay, A., Srinivasan, V., and Hasler, P., "A 531 nW/MHz, 128 x 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," in *Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004*, pp. 651 – 654, oct. 2004.

[18] Churchland, P. S. and Sejnowski, T. J., *The Computational Brain*. MIT Press, 1992.

[19] Delbruck, T., Koch, T., Berner, R., and Hermansky, H., "Fully integrated 500uw speech detection wake-up circuit," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 2015–2018, IEEE, 2010.

[20] Deng, Y., Chakrabartty, S., and Cauwenberghs, G., "Analog auditory perception model for robust speech recognition," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 3, pp. 1705–1709, IEEE, 2004.

[21] Diethorn, E., "A subband noise-reduction method for enhancing speech in telephony and teleconferencing," in *Applications of Signal Processing to Audio and Acoustics, 1997. 1997 IEEE ASSP Workshop on*, p. 4 pp., oct 1997.

[22] Ellis, R., Yoo, H., Graham, D. W., Hasler, P., and Anderson, D. V., "A continuous-time speech enhancement front-end for microphone inputs," in *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2002*, vol. 2, 2002.

[23] Farquhar, E. and Hasler, P., "A bio-physically inspired silicon neuron," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 3, pp. 477–488, 2005.

[24] George, S. and Hasler, P., "HMM classifier using biophysically based CMOS dendrites for wordspotting," in *Biomedical Circuits and Systems Conference (BioCAS), 2011 IEEE*, pp. 281 –284, nov. 2011.

[25] Georgiou, J. and Toumazou, C., "A 126-μw cochlear chip for a totally implantable system," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 2, pp. 430–443, 2005.

[26] Germanovix, W. and Toumazou, C., "Design of a micropower current-mode log-domain analog cochlear implant," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 47, no. 10, pp. 1023–1046, 2000.

[27] Gestner, B., Tanner, J., and Anderson, D., "Glass break detector analog front-end using novel classifier circuit," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pp. 3586–3589, IEEE, 2007.

[28] Gilbert, B., "A precise four-quadrant multiplier with subnanosecond response," *IEEE Journal of Solid-State Circuits*, vol. 3, no. 4, pp. 365–373, 1968.

[29] Gordon, C., Farquhar, E., and Hasler, P., "A family of floating-gate adapting synapses based upon transistor channel models," in *IEEE Internation Symposium on Circuits and Systems*, vol. 1, pp. 317–320, 2004.

[30] Graham, D. W., Farquhar, E., Degnan, B., Gordon, C., and Hasler, P., "Indirect programming of floating-gate transistors," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, pp. 951 –963, may 2007.

[31] Graham, D. W., Hasler, P. E., Chawla, R., and Smith, P. D., "A low-power programmable bandpass filter section for higher order filter applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 6, pp. 1165–1176, 2007.

[32] Gu, M. and Chakrabartty, S., "Synthesis of bias-scalable cmos analog computational circuits using margin propagation," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 2, pp. 243–254, 2012.

[33] Hall, T., Twigg, C., Gray, J., Hasler, P., and Anderson, D., "Large-scale field-programmable analog arrays for analog signal processing," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 11, pp. 2298–2307, 2005.

[34] Hasler, P., Diorio, C., Minch, B., and Mead, C., "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7* (Gerald Tesauro, D. S. T. and Leen, T. K., eds.), (Cambridge, MA), pp. 817–824, MIT Press, 1994.

[35] Hasler, P., Diorio, C., Minch, B., and Mead, C., "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7* (Gerald Tesauro, D. S. T. and Leen, T. K., eds.), (Cambridge, MA), pp. 817–824, MIT Press, 1994.

[36] Hasler, P. and Dugger, J., "Correlation learning rule in floating-gate pfet synapses," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 65–73, Jan 2001.

[37] Hasler, P. and Dugger, J., "An analog floating-gate node for supervised learning," *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 834–845, May 2005.

[38] Hasler, P. and Dugger, J., "An analog floating-gate node for supervised learning," *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 834–845, May 2005.

[39] Hasler, P., Minch, B., and Diorio, C., "An autozeroing floating-gate amplifier," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 74–82, Jan 2001.

[40] Hausser, M. and Mel, B., "Dendrites: bug or feature?," *Current Opinion in Neurobiology*, vol. 13, no. 3, pp. 372–383, 2003.

[41] Hertz, J., Krogh, A., and Palmer, R., *Introduction to the theory of neural computation*, vol. 1. Westview press, 1991.

[42] Hooper, M., Kucic, M., and Hasler, P., "Integration of high voltage charge-pumps in a submicron standard cmos process for programming analog floating-gate circuits," in *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 125–128, 2005.

[43] Hu, Y. and Loizou, P., "Subjective evaluation and comparison of speech enhancement algorithms," *Speech Communication*, vol. 49, pp. 588–601, 2007.

[44] i Petit, A. B. and Murray, A. F., "Synchrony detection and amplification by silicon neurons with stdp synapses," *IEEE Trans. Neural Netw.*, vol. 15, pp. 1296–1304, Sep 2004.

[45] Indiveri, G., "Modeling selective attention using a neuromorphic analog VLSI device," *Neural computation*, vol. 12, no. 12, pp. 2857–2880, 2000.

[46] Indiveri, G., "A current-mode hysteretic winner-take-all network, with excitatory and inhibitory coupling," *Analog Integrated Circuits and Signal Processing*, vol. 28, no. 3, pp. 279–291, 2001.

[47] Indiveri, G., Chicca, E., and Douglas, R., "A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *Neural Networks, IEEE Transactions on*, vol. 17, no. 1, pp. 211–221, 2006.

[48] Indiveri, G., Chicca, E., and Douglas, R., "A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Trans. Neural Netw.*, vol. 17, pp. 211–221, Jan 2006.

[49] Indiveri, G., Horiuchi, T., Niebur, E., and Douglas, R., "A competitive network of spiking VLSI neurons," in *World Congress on Neuroinformatics*, pp. 443–455, Vienna, Austria: ARGESIM/ASIM Verlag, 2001.

[50] Indiveri, G., Murer, R., and Kramer, J., "Active vision using an analog VLSI model of selective attention," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 48, no. 5, pp. 492–500, 2001.

[51] Itti, L., Koch, C., and Niebur, E., "A model of saliency-based visual attention for rapid scene analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 11, pp. 1254–1259, 1998.

[52] Kinget, P., "Device mismatch and tradeoffs in the design of analog circuits," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 6, pp. 1212–1224, 2005.

[53] Koch, C., *Biophysics of Computation*. Oxford University Press, 1999.

[54] Koch, C., *Biophysics of Computation*. Oxford University Press, 1999.

[55] Kruger, W., Hasler, P., Minch, B., and Koch, C., "An adaptive WTA using floating gate technology," *Advances in Neural Information Processing Systems*, pp. 720–726, 1997.

[56] Lazzaro, J., "Winner-take-all networks of o (n) complexity," tech. rep., DTIC Document, 1988.

[57] Lee, J., Bandyopadhyay, A., Faik Baskaya, I., Robucci, R., and Hasler, P., "Image processing system using a programmable transform imager," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP '05)*, vol. 5, 2005.

[58] Lim, J. S. and Oppenheim, A. V., "Enhancement and bandwidth compression of noisy speech," *Proceedings of the IEEE*, vol. 67, no. 12, pp. 1586–1604, 1979.

[59] Liu, B., Chen, C., and Tsao, J., "A modular current-mode classifier circuit for template matching application," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 47, no. 2, pp. 145–151, 2000.

[60] Liu, S.-C. and Mockel, R., "Temporally learning floating-gate vlsi synapses," in *IEEE International Symposium on Circuits and Systems*, pp. 2154–157, 2008.

[61] London, M. and H
"ausser, M., "Dendritic computation," *Annu. Rev. Neurosci.*, vol. 28, pp. 503–532, 2005.

[62] Lyon, R. F. and Mead, C., "An analog electronic cochlea," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1119–1134, 1988.

[63] M. Pankaala, M. L. and Hasler, P., "Compact floating-gate learning array with stdp," in *International Joint Conference on Neural Networks*, pp. 2409–2415, 2009.

[64] Maass, W., "On the computational power of winner-take-all," *Neural Computation*, vol. 12, no. 11, pp. 2519–2535, 2000.

[65] Mahowald, M., *VLSI analogs of neuronal visual processing: a synthesis of form and function*. PhD thesis, Citeseer, 1992.

[66] Maliuk, D., Stratigopoulos, H., and Makris, Y., "An analog vlsi multilayer perceptron and its application towards built-in self-test in analog circuits," in *On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International*, pp. 71–76, IEEE, 2010.

[67] Markram, H., "The blue brain project," *Nature Reviews Neuroscience*, vol. 7, no. 2, pp. 153–159, 2006.

[68] Markram, H., Lubke, J., Frotscher, M., and Sakmann, B., "Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps," *Science*, vol. 275, pp. 213–215, 1997.

[69] Marr, B., Degnan, B., Hasler, P., and Anderson, D., "Scaling energy per operation via an asynchronous pipeline," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –5, 2012.

[70] Martin, R., "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 5, pp. 504–512, 2001.

[71] Mead, C., *Analog VLSI and neural systems*. Addison-Wesley, 1989.

[72] Mead, C., "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[73] Minch, B. A., "A low-voltage mos cascode bias circuit for all current levels," in *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2002*, vol. 3, pp. 619–622, 2002.

[74] Minch, B. A., "Synthesis of static and dynamic multiple-input translinear element networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 2, pp. 409–421, 2004.

[75] Morris, T., Horiuchi, T., and DeWeerth, S., "Object-based selection within an analog VLSI visual attention system," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 45, no. 12, pp. 1564–1572, 1998.

[76] Nease, S., George, S., Hasler, P., Koziol, S., and Brink, S., "Modeling and implementation of voltage-mode CMOS dendrites on a reconfigurable analog platform," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2011.

[77] Ozalevli, E., Hasler, P., and Higgins, C., "Winner-take-all-based visual motion sensors," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, no. 8, pp. 717–721, 2006.

[78] P. Hafliger, M. M. and Watts, L., "A spike based learning neuron in analog vlsi," in *Advances in Neural Information Processing Systems 9,(p. 692)* (M. C. Mozer, M. I. J. and Petsche, T., eds.), (Cambridge, MA), MIT Press, 1997.

[79] Parikh, D. N., Ravindran, S., and Anderson, D. V., "Gain adaptation based on signal-to-noise ratio for noise suppression," in *Proc. IEEE Workshop Applications of Signal Processing to Audio and Acoustics WASPAA '09*, pp. 185–188, 2009.

[80] Peng, S.-Y., Tsao, Y., Hasler, P. E., and Anderson, D. V., "A programmable analog radial-basis-function based classifier," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing ICASSP 2008*, pp. 1425–1428, 2008.

[81] Peng, S., Hasler, P., and Anderson, D., "An analog programmable multidimensional radial basis function based classifier," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, no. 10, pp. 2148–2158, 2007.

[82] Petre, C., Schlottmann, C., and Hasler, P., "Automated conversion of simulink designs to analog hardware on an fpaa," in *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2008*, pp. 500–503, 2008.

[83] Rall, W., "Theoretical significance of dendritic trees for neuronal input-output relations," *Neural theory and modeling*, pp. 73–97, 1964.

[84] Rall, W. and Segev, I., "Functional possibilities for synapses on dendrites and on dendritic spines," *Synaptic function*, pp. 605–636, 1987.

[85] Rall, W. and Shepherd, G., "Theoretical reconstruction of field potentials and dendrodendritic synaptic interactions in olfactory bulb," *Journal of Neurophysiology*, vol. 31, no. 6, p. 884, 1968.

[86] Ramakrishnan, S., Basu, A., Brink, S., Chiu, L.-K., and Hasler, P., "Reconfigurable platform for implementing speech processing algorithms," *Circuits and Systems I, IEEE Transactions on*, p. Under Revision Review, 2011.

[87] Ramakrishnan, S., Hasler, P., and Gordon, C., "Floating gate synapses with spike time dependent plasticity," in *IEEE International Symposium on Circuits and Systems*, pp. 367–372, 2010.

[88] Ramakrishnan, S., Hasler, P., and Gordon, C., "Floating gate synapses with spike time dependent plasticity," in *IEEE International Symposium on Circuits and Systems*, pp. 367–372, 2010.

[89] Ramakrishnan, S., Hasler, P., and Gordon, C., "Floating gate synapses with spike-time-dependent plasticity," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 5, no. 3, pp. 244 –252, 2011.

[90] Ravindran, S., *Physiologically motivated methods for audio classification*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2006.

[91] Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[92] Sarpeshkar, R., *Efficient precise computation with noisy components: extrapolating from an electronic cochlea to the brain*. PhD thesis, California Institute of Technology, Pasadena, CA, 1997.

[93] Sarpeshkar, R., *Efficient precise computation with noisy components: extrapolating from an electronic cochlea to the brain*. PhD thesis, California Institute of Technology, 1997.

[94] Sarpeshkar, R., "Analog versus digital: extrapolating from electronics to neurobiology," *Neural Computation*, vol. 10, no. 7, pp. 1601–1638, 1998.

[95] Sarpeshkar, R., Baker, M., Salthouse, C., Sit, J., Turicchia, L., and Zhak, S., "An analog bionic ear processor with zero-crossing detection," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, pp. 78–79, IEEE, 2005.

[96] Satyanarayana, S., Tsividis, Y., and Graf, H., "A reconfigurable VLSI neural network," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 1, pp. 67–81, 1992.

[97] Schlottmann, C. and Hasler, P., "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 1, pp. 403 –411, sept. 2011.

[98] Schlottmann, C., Shapero, S., Nease, S., and Hasler, P., "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *Solid-State Circuits, IEEE Journal of*, vol. 47, pp. 2174 –2184, sept. 2012.

[99] Segev, I. and Rall, W., "Computational study of an excitable dendritic spine," *Journal of Neurophysiology*, vol. 60, no. 2, p. 499, 1988.

[100] Shapero, S. and Hasler, P., "Precise programming and mismatch compensation for low power analog computation on an FPAA," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, p. To be published.

[101] Siu, K., Roychowdhury, V., and Kailath, T., *Discrete neural computation: a theoretical foundation*. Prentice-Hall, Inc., 1995.

[102] Sivilotti, M., *Wiring considerations in analog VLSI systems, with application to field-programmable networks*. PhD thesis, Citeseer, 1990.

[103] Smith, P., Graham, D., Chawla, R., and Hasler, P., "A five-transistor bandpass filter element," in *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, vol. 1, pp. I–861, IEEE, 2004.

[104] Srinivasan, V., Graham, D., and Hasler, P., "Floating-gates transistors for precision analog circuit design: an overview," in *Circuits and Systems, 2005. 48th Midwest Symposium on*, pp. 71 –74 Vol. 1, aug. 2005.

[105] Srinivasan, V., Serrano, G. J., Gray, J., and Hasler, P., "A precision cmos amplifier using floating-gate transistors for offset cancellation," *Solid-State Circuits, IEEE Journal of*, vol. 42, pp. 280 –291, feb. 2007.

[106] Suh, S., Basu, A., Schlottmann, C., Hasler, P., and Barry, J., "Low-power discrete fourier transform for OFDM: A programmable analog approach," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, pp. 290 –298, feb. 2011.

[107] Twigg, C. M., Gray, J. D., and Hasler, P. E., "Programmable floating gate fpaa switches are not dead weight," in *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2007*, pp. 169–172, 2007.

[108] Twigg, C. M., Hasler, P., and Anderson, D. V., "Large-scale fpaa devices for signal processing applications," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing ICASSP 2007*, vol. 2, 2007.

[109] Urahama, K. and Nagao, T., "K-winners-take-all circuit with o (n) complexity," *Neural Networks, IEEE Transactions on*, vol. 6, no. 3, pp. 776–778, 1995.

[110] Widrow, B. and Winter, R., "Neural nets for adaptive filtering and adaptive pattern recognition," *Computer*, vol. 21, no. 3, pp. 25–39, 1988.

[111] Yamasaki, T. and Shibata, T., "Analog soft-pattern-matching classifier using floating-gate mos technology," *Neural Networks, IEEE Transactions on*, vol. 14, no. 5, pp. 1257–1265, 2003.

[112] Yildiz, M., Minaei, S., and Göknar, İ., "A flexible current-mode classifier circuit and its applications," *International Journal of Circuit Theory and Applications*, vol. 39, no. 9, pp. 933–945, 2011.

[113] Yldz, M., Minaei, S., and Goknar, I., "A cmos classifier circuit using neural networks with novel architecture," *Neural Networks, IEEE Transactions on*, vol. 18, no. 6, pp. 1845–1850, 2007.

[114] Yu, T. and Cauwenberghs, G., "Analog VLSI biophysical neurons and synapses with programmable membrane channel kinetics," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 4, no. 3, pp. 139–148, 2010.

VITA

Shubha Ramakrishnan received her B.E. in Electronics from Birla Institute of Technology and Science, Pilani, India and her MS in Electrical Engineering from Oregon State University in 2002 and 2004 respectively. She received the PhD degree in Electrical and Computer Engineering at Georgia Institute of Technology. Her interests include low power analog design for signal processing and classification, bio-inspired circuit design, and modeling biological learning processes in silicon.