HARDWARE ACCELERATION OF NEURAL NETWORKS WITH CMOS AND POST-CMOS DEVICES

GOVIND NARASIMMAN

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2017

HARDWARE ACCELERATION OF NEURAL NETWORKS WITH CMOS AND POST-CMOS DEVICES

GOVIND NARASIMMAN

School of Electrical & Electronic Engineering

A Thesis Submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Master of Engineering

2017

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Arindam Basu, for his patient guidance and helpful advices, without which this thesis could never have been completed. I am grateful to Assoc. Prof. Chang Chip Hong for giving me an opportunity to work and learn in this interdisciplinary project. I also acknowledge the financial support for this work, which was provided by Ministry of Education, Singapore through grant MOE2013-T2-2-017. I am thankful to research scholars Dr. Zhang Le , Dr. Subhrajit Roy. Mr Pankaj Sethi for their valuable suggestions. I would also like to acknowledge the useful discussions and materials provided by our collaborators at Purdue University - U.S.A, Prof. Kaushik Roy and his group members Asst. Prof. Mrigank Sharad and Asst. Prof. Xuanyao Fong.

I would also like to thank the Virtus IC Design Centre of Excellence-EEE for providing me with all the facilities for my research work. I am thankful to all my group members for their helpful inputs and discussions on the subject. I would also like to thank Virtus Research scholars Rahul, Arish ,Roshan ,Vigil for their inspirational and positive support through-out the research. I am very grateful to my parents for being supportive through the course of study at NTU. Lastly, I am thankful to my brother and fellow researcher Neelakantan for his emotional support and guidance for completion of the research work.

Abstract

There is a huge need for embedded machine learning for portable devices and smart sensors to power the next generation of Internet of Things (IoT). Implementation of neural networks involve large number of arithmetic and memory operations. Realization of the arithmetic blocks with conventional digital circuits will inherit a trade-off between accuracy of calculation and area required. On the other hand, existing analog building blocks for neural networks suffer from inaccuracies related to process variation and large power consumption. Moreover, the efficiency of computation-memory interface is degrading since memory bandwidth increment is poor compared to computation throughput growth in CMOS technology. Though the recent large scale neuromorphic circuits have used localized random access memory for reducing memory operations, this local memory size will not be scalable with increasing size of datasets. Here, we explore novel CMOS and post-CMOS circuits to realize ultra-low power neuromorphic circuit by co-design of algorithm and hardware. The solutions also overcome the issue of increasing bandwidth gap between memory operations and computation.

First, a deep neural network with 2 convolution layers and 2 fully connected layers, is chosen and tuned for hardware implementation. The network has few tunable parameters (≤ 40000) and is 40 times faster in training than usual deep neural networks with 4 layers. We propose a compact, single transistor element for realizing the connections inside the neural network- called 'synapse'. These synapses perform

the computations involved by virtue of mismatch inherent in their fabrication. We use a current mirror array with n -input lines and m output lines, to perform ' $n \times m$ ' Multiply and Accumulation operation. The resultant neuromorphic circuit can emulate multi-layered artificial vision system. A circuit fabricated in $0.35\mu\text{m}$ CMOS process is characterized and a behavioral model is simulated for the deep neural network. Here, the learning is done offline. The inputs to the network may vary according to environmental conditions, for which we need an adaptive neural network. Hence we propose a second solution where the neuromorphic circuit can adapt its parameters in real time.

With the advent of novel nanoscale devices with physical properties well matched to neural network enabling computations at energies much lower than CMOS, the research also focuses on use of a post-CMOS spintronic device-a Domain-Wall magnet for obtaining a low power spike timing dependent plastic (STDP) synapse for online learning. The spin-mode signals are injected across small potential ($\approx 50\text{mV}$) through multiple layers of ferromagnetic and non-magnetic layers. Here we discuss the implementation of a spiking neural network with synapses which can be trained according to STDP learning rule. A detailed study with the help of device circuit co-simulation is done. Possible use of this synapse in online, real time learning spiking neural networks is also illustrated in this thesis.

Contents

Acknowledgement	i
Abstract	iii
List of Figures	viii
List of Tables	xvi
List of Abbreviations	
1 Introduction	1
1.1 Neural Network	2
1.2 Motivation and Objective	3
1.3 Organisation of the Thesis	5
2 Hardware Acceleration of Artificial Neural Network: A CMOS solution	7
2.1 Background of Neural Network Algorithms	8
2.1.1 Convolutional Neural Network	8
2.1.2 Extreme Learning Machine	14
2.2 Literature Survey : Hardware Implementation of CNN	15
2.2.1 Hardware implementation of Random Projection Core	19
2.3 A Low Power Hardware Implementation of CNN	22
2.4 Results	25
3 Post-CMOS Solution for Self-Learning Low-power Synapse	30

3.1	Background: Spiking Neural Network (SNN) and Spike Timing Dependent Plasticity (STDP)	31
3.2	Literature Survey of Self-Learning Synapses for SNN	33
3.2.1	Self-learning Synapse in CMOS	33
3.2.2	Floating Gate based Self-learning Synapses	34
3.2.3	Memristor based Self-learning Synapse	36
3.3	Enabling STDP with DW based Self-learning Synapse for SNN	37
3.3.1	Introduction to Spin based devices	37
3.3.2	Spin based Neuron module for Non-Boolean Computation	38
3.3.3	DW Motion Characteristics	40
3.3.4	Enabling STDP with DW synapse	45
3.3.5	System Integration	47
3.4	Results	50
3.4.1	Tunable STDP curves with Change in Circuit Parameters	50
3.4.2	Network Simulation: Handwritten Digit Classification	51
4	Algorithm for Improvement of Spin-mode Neuromorphic Circuit	56
4.1	Structural plasticity Algorithm	58
4.2	Experiment and Result	59
5	Conclusion and Future work	61
	Appendix	64
.A	Simulation Setup Used for Spin-Mode Neuromorphic Circuit	64
.B	Calculation of Write Energy per Spike	67

Publication	68
Bibliography	69

List of Figures

1.1	Prediction of number of devices connected to network [1]: For the past few years apart from conventional computers and tablets, sensors for personal health monitoring are extensively connected to network. In future the range of devices connected to Internet will include modules for home automation, smart automobiles to all household items. The trend plotted here suggests the need of "edge-computing" where meaningful information is extracted from these sensors before transmitting the data into the network.	2
1.2	Forward-path in basic neural networks. A neuron sums up the weighed inputs and subject the sum to non-linearity $G()$	3
2.1	Sequence followed in classification : Discrete feature extraction facilitates easy classification with lower dimensional feature	7
2.2	illustration of Convolution: a) In convolution stage, filters are moved across images/feature maps to produce output feature map. b)Each filter produces a high response in feature map, if the feature embedded in the filter is present in input image [2].	9

- 2.3 **Random Combination of Pooled second layer features** : The third layer features are obtained by combining 'V' number of feature maps generated with different filters from distinct second layer maps. The combination of second layer features for producing third layer features is represented further by the symbol on right side. 11
- 2.4 **A two stage CNN** : The figure shows layers involved in a 2-stage CNN . Note that third layer features are formed by combination of convolved output of 'V' filters with 'V' second layer features. LCN is performed on first layer features ,but it does not change size or count of features. . . 12
- 2.5 **ELM structure with single hidden layer:** In such SLFNs implementations,the activation function $g()$ for additive nodes can be any bounded non constant piecewise continuous functions. $g : R \rightarrow R$. In this figure there are c output classes. 14
- 2.6 **An internal diagram of collection:** Collections are core configurable computation engines of nn-X. It performs data routing, convolutions, pooling, non-linear programmable functions. Purple lines below are configuration buses [3]. 16
- 2.7 **Basic components of HAGEN chip** : Sign bit determines, whether to sink the input current from I_{pos} or I_{neg} . The neuron is a simple comparator of negative and positive currents 19
- 2.8 **The random projection core circuit:** Inputs are fed to current array with help of current DAC elements [4]. The input current gets mirrored in all grid elements. All current mirrors in a column draw total current I_{in} from one Silicon neuron. Thus the desired MAC operation response is generated in respective counters. CLK_{OUT} is used for shifting counter outputs out in 13 bit bus. The implementation in [4] consisted of 128 input channels preferably enabled with address line A < 0-7 >. 21

- 2.9 **The neuron used for pulsed output generation:** a) schematic of the current mode neuron circuit used in [4] b) The counter output of the CCO-ADC varies according to dotted line shown. Neuron spiking frequency initially increases with the increase of the input current I^{in} , until $I^{in} = I^{rst}/2$. It then reduces and becomes zero finally when $I^{in} = I^{rst}$ 22
- 2.10 **Proposed solution RP²CNN** . Input patch under consideration is rearranged as a vector and are shifted to input registers. Output responses of two consecutive CCO-ADC are subtracted to form one pixel output in output feature map. A cyclic pairwise subtraction can earn FM¹ feature maps. By scanning the input image once we can produce FM¹ feature maps. 23
- 2.11 **Processing stages were Random projection core can be used.** An efficient pipeline of the deep network is proposed as shown. Pooling layer outputs are generated with simple digital shift registers and adders. The first stage of ELM classifier can be implemented with the same random projection technique. Finally the output classes are obtained from preloaded output weight β 23
- 2.12 **Rotation of inputs field:** Rotation of bits results in each input field being multiplied by new random vector, hence producing distinct feature map. 24
- 2.13 **Architecture of complete image recognition system.** All digital processing stages can be implemented inside a DSP peripheral block. The inputs of random-projection core are arbitrated for maximum utilization. The connection matrices of CNN stages are stored in routing stages for optimized for maximum throughput of pipeline. 25

2.14 Test setup used for characterizing weights of current mirror array:	
a) Inputs to each row is transfered into the shift register through a serial link from a 'Spartan 3' FPGA. After obtaining the output CCO responses ,they are shifted into the FPGA and transfered to PC. b) Die photograph of the current mirror array fabricated in $0.35\mu\text{m}$ CMOS technology. . . .	26
2.15 Distribution of weights in the chip: a) Figure shows distribution of Log-normal weights observed by virtue of mismatch in the chip. b) A zero-mean distribution of weights obtained for feature detection by subtracting adjacent column counter outputs.	27
2.16 First layer feature maps: Figure contains first layer features maps of inputs a) Digit eight. b) butterfly. and c) camera. Different feature responses are seen high in different maps.	29
3.1 STDP characteristics from [5] is shown. t_{pre} and t_{post} corresponds to time instances of spike events of pre-synaptic neuron and post-synaptic neuron respectively.	32
3.2 Synapse circuit from [6]: The CMI circuit implements a current-mirror-integrator and generates a post-synaptic current that is injected into the neuron. The STD circuit necessarily acts as a high pass filter of inputs spikes weighing the initial stimulus more than subsequent input activity. The STDP based learning is implemented in the synapse with the help of memory traces V_{pot} and V_{dep} . This circuit, that drives the synaptic weight to one of two possible states on long time scales.	33
3.3 Floating gate transistor and terminal waveforms : a) The FG transistor's terminals with corresponding waveforms for LTD and LTP [7]. b) The system configuration adopted for Program mode.	35

- 3.4 **Lateral spin valve** : The vertically inserted input charge current generates a spin polarized current in the channel. The charge current in the channel flows into the large ground lead whereas the spin polarized current is absorbed by the small output magnet. Ground lead can be biased at a voltage close to power-supply for providing adequate input current. Hence 'non-local spin transfer torque' was exploited several ultra low-power boolean and non-boolean applications [8]. 38
- 3.5 **DW magnet strip as synapse** a) Position of DW magnet w.r.t center of the strip becomes of weight W of the DW-synapse. The symbol shown in right side will be used for further representation. b) Spin injection strength into channel for different charge current X_{in} as a function of W [9]. 39
- 3.6 **Sequence of events in MTJ output evaluation**; [9]. a) A preset pulse pushes the free-layer magnet orientation (M_z) to meta stable state. As soon as the preset pulse is removed (I_{net}^{spin}) decides the orientation of free-layer magnet. Clock pulse is synchronized with the preset pulse for MTJ o/p shown in Fig. ??(b). b) A preset magnet is added in the structure along with DW magnets shown in Fig. ?. The magnetization of preset magnet is directed orthogonal to output free-layer magnet [9]. . . 41
- 3.7 **Displacement achieved with different current pulse amplitudes** a) Width of pulse used here is 100ns b) Final position of Domain wall normalized w.r.t notch interval , on application of current pulse with amplitude shown in x-axis. Maximum and minimum limits of weight W will be 8 and -8 respectively, limited by the number of notches included. . . . 42
- 3.8 **Cross talk current I_{CT}** : The stray spin current from one DW magnet may enter another adjacent DW magnet and disturb the domain wall position and hence weight too. 44

- 3.9 **decrease in I_{CT} with increasing channel thickness:** increasing t_{ox} increases interface resistance. T_{ch} is thickness of non-magnetic channel. 45
- 3.10 **Write circuitry of DW Magnet synapse** a) Write current generation from sampling of $T_{pre(post)}$ by $S_{post(pre)}$. At instance t_1 , a post synaptic spike S_{post} happens which samples T_{pre} waveform. The resulting write current creates a positive ΔW according to Δt . On the other hand, a negative current is generated by a pre-synaptic spike S_{pre} , at time instance t_2 , to decrease the weight 'W' of the corresponding synapse. b) Generation of $T_{pre/post}$ according to the memory trace. c) PMOS stack (M_3 - M_6) used to generate the write current to obtain the desired ΔW . V_0 and Δv are 750mV and 50mV respectively. d) Compact representation of the whole DW magnet based synapse completed with the write circuit. The deep triode current source transistor M_1 provides input charge current I_r 46
- 3.11 **System integration** a) Clocked differential latch employed to detect MTJ output change. b) Signal flow in neuromorphic circuit: Inputs $X_{in}(t)$ are propagated through the neuromorphic circuit to produce synchronous current pulses $I_{in}(t)$. This $I_{in}(t)$ is integrated by I&F neuron to produce asynchronous spikes. 48
- 3.12 **Integrate and fire Neuron and membrane potential evolution** a) Time evolution of membrane potential V_m (from SPICE simulation) as a result of clocked MTJ output evaluation (see Fig. ??(a)). The ACK signal resumes the integration of total input current $I_{in}(t)$ by I&F neuron. b) Integrate and fire neuron from [10]: $I_{in}(t)$ is the total input current to I&F neuron. The leak current of the I&F neuron can be tuned by potential V_{LK} . Hence V_{LK} and capacitor C will decide τ_m in equation (??). ACK and REQ are handshake signals to facilitate large number of connections 49

3.13 tunable STDP curves a) STDP curve obtained from SPICE simulation of the circuit shown in Fig. ??.	
b) ΔW_{max} increases with I_{max} , where I_{max} is the write current corresponding to $\Delta t = 0$.	
c) STDP curve obtained for ratio of time windows $(T_{pre}/T_{post}) \in < 1$.	
d) R-STDP curve obtained by interchanging T_{pre} and S_{post} with T_{post} and S_{pre} respectively in Fig. ??(c)	51
3.14 Network architecture used for simulation. T is the time duration of one pattern.	53
3.15 Weight update rule for semi-supervised learning: The weight update for the synapses of neuron N_n when it is (a) supposed to respond to the current pattern but it remains silent, (b) supposed to remain silent but responds and (c) supposed to respond and it does so. The first two figures show how our learning rule handles error and the third figure shows how homeostasis or stop-learning is achieved.	53
3.16 Receptive fields :Receptive fields obtained for hand written digits 0 and 1, after training with network architecture shown in Fig. ??.	55
4.1 Layout of neuromorphic circuit : Increasing number of synapses per branch of neural network would reduce the spin current available beneath MTJ free-layer.It can be noticed from the top-view of the synapse-neuron module, that decreasing length of DW-Magnets will be also useful to avoid the bottleneck of shorter spin-flip length.	57
4.2 Testing accuracy of network: The result shows increase in network performance with increase in number of perceptrons per cell (m). Also the results show improvement in performance with higher number of bits in synapse. the difference in testing accuracy diminishes for larger m.	60

.1	Simulation framework : Simulation framework employed in this work is similar to the one in [9]. Block (1), which mimics the DW motion under the influence of current pulse is explained by equation (??) and (??). The same is simulated using Verilog A model.	64
.2	Multi-input LSV structure [11]: Magnetization dynamics and spin-transport model of all ferromagnets and magnetchannel interface will be self-consistently simulated. The magnetization of the input ferromagnets (partitioned by DW) will be obtained from 1-D model as explained in section ?? . The model shown is interpolated to 2-D plane as explained in [9].	65
.3	π Conductance model : Pi conductance model for G_{FM-NM} , G_{FM} and G_{NM}	66

List of Tables

2.1	Network Parameters used for different datasets	13
2.2	Comparison of two random feature extraction methods.	27
2.3	Testing accuracy of the network in Matlab simulation.	28
3.1	Magnetic parameters of DW strip.	43
3.2	Comparison of Write Energy/spike	52

List of Abbreviations

ANN	Artificial Neural Network	SNN	Spiking Neural Network
CNN	Convolution Neural Network	FNN	Feed-forward Neural Network
STDP	Spike timing dependent plasticity	CNN	Convolution Neural Network
R-STDP	Reverse-Spike timing dependent plasticity	LTP	Long term potentiation
STD	Short term depression	LTD	Long term depression
RAM	Random Access Memory	CMI	Current mirror integrator
CMOS	Complementary Metal Oxide Semi-conductor	FG	Floating Gate
MTJ	Magnetic Tunnel Junction	DW	Domain Wall
PMA	Perpendicular Magnetic Anisotropy	TMR	Tunnel Magneto resistance
CCO	Current Controlled Oscillator	ADC	Analog to Digital Converter
MgO	Magnesium Oxide	DAC	Digital to Analog converter
RPC	Random projection Core	LCN	Local Contrast Normalisation
SLFN	Single Layer Feed-forward Network	ELM	Extreme Learning Machine
STT	Spin Transfer Torque	ASL	All Spin Logic
LSV	Lateral Spin Valve	DTCS	Deep Triode Current Source

Chapter 1

Introduction

With the emergence of Internet of Things (IoT), sensor nodes connected to the network are increasing exponentially. The prediction of number of sensors devices connected to network is shown in Fig. 1.1. The data transmitted by these sensor nodes will create huge traffic for communication with any remote server. This has led to development of a new model for communication and analysis of data from these sensors- Edge computing. In Edge computing, meaningful information is interpreted before delivering the data into the network by a low power cognitive node. Artificial Neural-Networks have been the basis for powerful algorithms with applications in cognitive tasks like computer vision, robot kinematics and pattern-recognition. Efficient implementation of artificial neural-network (ANN) architectures is key to achieve human like performance in cognitive tasks. Computation and data transfer requirements of such ANN cannot be met with conventional instruction set based processors. A compact solution for such applications when implemented with conventional digital circuits will consume large area [12]. A specific and custom-made hardware is necessary to decrease execution time and achieve low-power consumption in such tasks.

CHAPTER 1. INTRODUCTION

Figure 1.1: **Prediction of number of devices connected to network [1]:** For the past few years apart from conventional computers and tablets, sensors for personal health monitoring are extensively connected to network. In future the range of devices connected to Internet will include modules for home automation, smart automobiles to all household items. The trend plotted here suggests the need of "edge-computing" where meaningful information is extracted from these sensors before transmitting the data into the network.

1.1 Neural Network

Artificial Neural Networks: ANN can be divided into two main categories: *feed-forward networks* (FFN), in which the computation is performed in a layer-by-layer fashion from the input to the output of the network; and *recurrent networks* which have an interconnected network structure including cycles. The general structure of all ANN are the same, in analogy with biology they are made up of 'neurons' and 'synapses'. Synapses are the primary wiring component between neurons. Their number increases quadratically with number of neurons. They have associated weights, which should be medium to high precision for fast convergence of learning algorithm. Each synapse weighs the input applied through them according to their respective weight.

$$Y = G\left(\sum_i^N X_i W_i\right) \quad (\text{Eq. 1.1})$$

Figure 1.2: **Forward-path in basic neural networks.** A neuron sums up the weighed inputs and subject the sum to non-linearity $G()$.

For online learning, these synaptic weights have to be updated during regular operation. Neurons perform summation of weighted inputs applied through synapse. They also calculate non-linear activation function G of the sum obtained. In this work, we focus on FFN (see Fig. 1.2).

Spiking neural networks (SNN) : SNN consist of spiking neurons, which unlike the ANN of previous generations, do not generate outputs on each evaluation cycle. Instead, the neurons produce a spike asynchronously when an internal quantity, membrane potential (V_m), reaches a specific value. Hence, the information transfer in SNN takes place through precise timing of spike or rates of spikes. Such SNN are being widely studied due to their strong resemblance to biology [13], as well as due to its stronger computational properties [14]. In terms of hardware realization, these also have advantage of dissipating power based on activity since only spikes or events need to be communicated between neurons [15]. Also, address event representation (AER) protocols allow for arbitrary configuration of network topology by using separate memory modules to store connection tables [16].

1.2 Motivation and Objective

With the emergence of very deep neural network in the recent past, image recognition architectures with 60 million synapses and tunable parameters like [17] are com-

CHAPTER 1. INTRODUCTION

mon. Since number of synapses in the network are ($100\times$ typical) more than neurons, synapses should be compact and efficient. There are several CMOS implementation of synapses in literature. Digital implementation of ANN requires complex arithmetic units and memory elements. This requires large area and hence impedes vast integration.

Storage of synaptic weights in analog CMOS implementation of neural networks is limited to two states, high and low in long term [6]. This is due to imperfections in fabrication like tunneling and leakage. Synapses can also be implemented by storing weights in a digital register, as is done for the digital synapse. Performing the computation, will then require analog to digital converter (ADC)/ digital to analog converter (DAC) elements per synapse used in the network. This will cause a delay in computation. Also the circuit thus implemented would have to trade-off the power and area scaling with precision given to the synaptic weights [18]. Moreover the design of analog circuits cannot be automated as much as their digital counterparts. These reasons drive us to explore more robust analog/mixed signal implementation of synapse for use in deep networks. In next Chapter we concentrate on developing compact single transistor synapse compatible for tasks like image recognition.

Use of Non-volatile Memory synapses can help us to design compact and easily accessible synapse for self-learning neuromorphic circuits. Many existing post-CMOS solutions have used floating gate (FG) voltage [19] or programmable memristor [20] to implement weight associated to synapse. Both FG devices and memristors have high voltage requirements for programming the underlying weight and hence reduce the reliability of the circuit and increase the energy needed for on-chip learning.

Recently, there is increased interest in research of possible devices that specifically exploit spin properties instead of or in addition to charge degrees of freedom. Due to spin-based interactions between the ferromagnets and electrons, the magnetic orientation of ferromagnetic elements can impart spin polarization to current flowing through it. By means of these same interactions, the spins of electron entering the fer-

CHAPTER 1. INTRODUCTION

romagnet can also influence the orientation of magnetization of the ferromagnet. This helps in storage of non-volatile information in ferromagnets by virtue its magnetization direction. A magnetic tunnel junction (MTJ) is generally a stack of two ferromagnets separated by a thin Magnesium Oxide (MgO), that can be used to convert the information stored in magnetic domain to electrical domain. The effective conductance of the stack is lower when magnetic moment of both ferromagnetic layers (free-layer and a thick reference layer) are aligned parallel. The conductance of the stack is higher when magnetic moment are anti-parallel to each other. The critical charge current required to reverse the free-layer magnetization can be reduced extensively with clocking schemes similar to [21].

The achievement of significant Tunnel Magneto resistance (TMR), in metallic Magnetic tunnel junctions with (MgO) as tunneling barrier [22], have turned spin based modules as a preferred candidate for non-volatile memory in low power applications. Magnetic domain wall is an interface separating the regions which have uniform magnetization. Within the magnetic domains, the individual magnetic moments of the atoms are aligned with one another and they point in the same direction. When the size of a magnetic material is reduced, the number of domains and domain walls can be controlled systematically. In case of nanoscale materials, the motion of domain wall can be controlled in particular direction and study their dynamics for the development of futuristic novel magnetic storage devices.

There are several studies which show that DW motion is possible with low current densities as low as 10^{-11} A/m^2 [23]. These factors motivate us to use spin-based signaling and nano-magnetic devices for implementation of simple neural network structures.

1.3 Organisation of the Thesis

The report is organized as follows. In Chapter 2, we discuss a power efficient and simple method for feature extraction from images. We propose an analog/mixed signal

CHAPTER 1. INTRODUCTION

co-processor to obtain compact and distinct random features of input image. We also show through simulations a feasible, end to end recognition of images, with the help of these analog/mixed signal co-processor and digital peripherals. The learning here is done offline.

In Chapter 3 we discuss a compact, low power and low-voltage self learning synapse with ferromagnetic nano-strips. Here, we discuss a neuromorphic circuit which uses spin mode signaling for implementation of a SNN. The DW Magnet-based synapse involved , consumes very low-energy/spike in learning it's synaptic weight.

In Chapter 4 we discuss a design space exploration for optimal structure of the neuromorphic circuit. This optimization will alleviate the limitations of spin-based signaling in neural network. Finally, Chapter 5 presents the conclusion and future discussion.

Chapter 2

Hardware Acceleration of Artificial Neural Network: A CMOS solution

Applications of Convolutional Neural Networks (CNN) range from recognition (such as handwriting recognition, facial expression recognition and video surveillance) and reasoning to intelligent text applications such as semantic text analysis and natural language processing applications. A CNN is comprised of one or more convolutional layers combined with pooling layers. The CNN is useful in extracting differentiable features of the input image. These feature vectors can be classified by any other independent classifier. CNN is followed by one or more fully connected layers as in a standard multilayer neural network. Here, more stress is given to computer vision

Figure 2.1: **Sequence followed in classification** :Discrete feature extraction facilitates easy classification with lower dimensional feature

based application owing to large data involved in computation. The architecture of a CNN is designed to take advantage of the 2-D structure of an input image. The

goal of the chapter is to present **a) robust mixed signal CMOS solution, for feature extraction in CNN. b) to effectively use the inherent device mismatch of VLSI fabrication in implementation of this feature extraction co-processor.**

In the convolution layer of neural network connections to next layer are restricted to small subset of image. The weights thus form an image filter. The image filter is traversed across the image with a stride of one pixel. Inputs such as images have strong 2-D correlation. Local correlations are the reason for the well known advantage of extracting and combining local features, before recognition of spatial objects, because configuration of neighboring pixels in an image can be classified into small categories [24], such as edges, slants ,corners etc. For problems like image recognition, traditional multi layer perceptron (MLP) like networks will involve learning of huge amount weights and associated storage, but the localized connection and weight sharing in CNN reduce number of connections. Therefore CNN based architecture is suitable for hardware implementation also.

2.1 Background of Neural Network Algorithms

2.1.1 Convolutional Neural Network

This section discusses each block of CNN briefly.

Convolution Convolution of an input image with filter corresponds to scanning of an image for a feature. Each neuron takes inputs from a rectangular section of the previous layer. The weights for rectangular sections are the same for each neuron in the next layer. Thus, the convolutional layer is just an image convolution of the previous layer, where the weights specify convolution filter. Distinct filter kernels produce distinct feature maps (see Fig. 2.2). The output after convolution can be written as:

$$Y_{x,y}^{i,S+1} = G\left(\sum_{m,n} X_{x+m,y+n}^{i,S} W_{m,n}^{i,S}\right) \quad (\text{Eq. 2.1})$$

where G is the non-linearity applied, i is feature map index , S is layer number x and y give spatial location of output on feature map and m,n gives the spatial neighborhood

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.2: **illustration of Convolution:** a) In convolution stage, filters are moved across images/feature maps to produce output feature map. b) Each filter produces a high response in feature map, if the feature embedded in the filter is present in input image [2].

of convolution for filter. By taking into account the borders effect, we have the size of output feature map $x^{S+1} = x^S - m^S + 1$, where x^{S+1} is size of next layer and m^S is filter size. Several non-linearity like tanh/sigmoidal are used in literature for squashing the feature output. Analysis in [2] [25] have found Rectified linear unit(ReLU) and absolute value non-linearity to be more effective. A ReLU type non-linearity can be implemented with truncation of sign bit. Absolute value (abs) is another non-linearity commonly employed. These replicated units share the same parameterization (weight vector and bias) and form a feature map. The feature maps can respond to simple filters, where responses will be recorded to specific edge like filters. We can also learn these filters in an unsupervised manner (with/without supervised fine tuning) and also in supervised method. [25] shows detailed comparison of the above said methods and provides useful inferences from the results they obtained. They also discuss performance of deep networks with fixed random filters, followed by a classifier trained in supervised manner.

Pooling Layer A pooling layer follows the convolution layer in CNN. Spatial Pooling not only reduces the data passed onto next stage, but also makes the features in-

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

variant to small translations and rotations. Sub-sampling is one kind of pooling where average value of a small grid of pixels are replaced in resultant output layer. These grids are mostly non-overlapping. In Max-pooling, instead of average, maximum value input is passed to output layer. Size of the pooling layer varies according to size of input image and subsequent map sizes.

Local contrast normalization (LCN) Local Contrast Normalization enforces a local competition between adjacent features in a feature map, and between features at the same spatial location in all feature maps in one layer of CNN. This is achieved by subtractive normalization followed by a divisive normalizations. The subtractive normalization operation for a given pixel $X_{i,x,y}$ computes:

$$v_{i,x,y} = X_{i,x,y} - \sum_{i,p,q} W_{p,q} \cdot X_{i,x+p,y+q}, \quad (\text{Eq. 2.2})$$

where $W_{p,q}$ is a Gaussian weighting window normalized so that $\sum_{i,p,q} W_{p,q} = 1$. Note that i is feature map index and x,y are spatial index of pixels in feature map. The divisive normalization computes

$$y_{i,x,y} = v_{i,x,y} / \max(c, \sigma_{x,y}), \text{ where } \sigma_{x,y} = \left(\sum_{i,p,q} W_{p,q} \cdot v_{i,x+p,y+q}^2 \right)^{1/2} \text{ and } c=1. \quad (\text{Eq. 2.3})$$

Architecture of Deep CNN By cascading convolution layers and pooling layers we can make deep CNN. One CNN stage is a combination of convolution layer followed by Local contrast normalization (LCN) and pooling layer. CNN increase number of feature maps and decrease resolution of features in each map after each stage. A 8-layer deep CNNs can be formed by cascading several such stages as shown in [17]. The hierarchically repetitive structure, local simple features (e.g., line segments) in the input image are gradually assembled into complex features. CNN develop spatial invariance to features and specificity of features in distinct feature maps. A two stage CNN with a supervised classifier found in [2] [25] are proven to efficient in classification

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

of image dataset MNIST [26], CALTECH101 [27] and NORB dataset. Following are the steps involved in a two stage CNN (see Fig.2.4).

An image is subjected to simple pre-processing steps before introducing it into the CNN. Pre-processing steps include 1) Conversion of the image to gray scale and 2) Resizing is done by zero padding shorter dimension of image to desired length. 3) Subtracting the images mean and dividing by the image standard deviation. After first convolution layer, FM^1 feature maps are produced by convolution with FM^1 image filters. A LCN across feature maps of that layer can provide better optimization, if included. Note that LCN does not change size of feature map or number of feature maps. A sub-sampling layer achieves detection of features tolerant to spatial shifts. A second convolution layer (FM^2) produces FM^2 complex feature maps by combining 'V' convolved outputs with the help of $V * FM^2$ filter banks (see Fig.2.3). The number $V * FM^2$, can be reduced with help of an optimized connection matrix of feature maps and second stage filters. Fourth layer is sub-sampling layer (S^2) which reduces the

Figure 2.3: **Random Combination of Pooled second layer features** : The third layer features are obtained by combining 'V' number of feature maps generated with different filters from distinct second layer maps. The combination of second layer features for producing third layer features is represented further by the symbol on right side.

size of feature maps by further d^2 factor. A re-arranged fourth layer can be fed to any classifier to deduce the output class. Number of first stage outputs and size of image filters is dependent on size and type of dataset. Commonly used image filter sizes and number of first stage filter and second stage filter are given in Table 2.1.

Figure 2.4: **A two stage CNN** : The figure shows layers involved in a 2-stage CNN . Note that third layer features are formed by combination of convolved output of 'V' filters with 'V' second layer features. LCN is performed on first layer features ,but it does not change size or count of features.

Computation involved in CNN As shown in Eq. 2.1 convolution mostly involves multiply and accumulate (MAC) operation. For e.g. Convolution with $m_1 \times m_1$ filters involves x^2 MAC operation with m_1^2 inputs, where $x \times x$ is the input image/feature map size. Here $m_1 \times m_1$ is the size of convolution filter. This amounts to 784 MAC operation for each feature map in first stage of MNIST-CNN (see Table 2.1). The additions in pooling layer will only amount to a fraction of image/feature size. Another complex procedure in CNN is LCN. LCN are not necessarily needed in all stages of convolution unlike pooling. A local window based mean is calculated per pixel across all features in the stage. It is fair to assume the size of normalization window same as that of convolution kernel of corresponding stage. So the subtractive normalization also involves x^2 MAC operation. We also need to perform x^2 subtractions for each feature map. Divisive normalization also involves same complexity in calculating a variance

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Table 2.1: Network Parameters used for different datasets: According to size of input images and features present in images the size of filter varies. It is good to increase number of feature maps generated in each layer, but complexity in learning huge number of parameters can reduce small increments in accuracy.

Dataset	First stage filter size (l_1)	Second stage filter size (l_2)	# of First stage filters (FM^1)	# of second stage filters (FM^2)	# of features combined in third layer (V)
Caltech101	9	9	64	256	16
MNIST	5	5	50	128	10
Cifar10	7	5	50	128	12

across features as shown in equation Eq. 2.3. The complexity in normalization function can quickly reduced by approximating the window co-efficients to powers of 2. This reduces digital multipliers and divider circuits required.

Capability of Random Weights From [25] [2], it is known that random kernels (i.e. image filter without learning) followed by a supervised classifier can classify MNIST, Caltech with good accuracies. The following explanation from [28] is supported with mathematical analysis. To understand the type of input features preferred by CNN architectures, it would be useful to know the set of inputs that maximally activate the feature map. The *Feature Selectivity* of the architecture will be approximately similar to that of the filter W used in the convolution layers [28].

For a highly frequency selective filter W , a sinusoidal input with frequency close to that of W is optimal. However optimal inputs for a random filters are not diffusive or random. From results of [28] we know that, regardless of filter W optimal inputs to

stimulate a filter is a sinusoidal input of frequency maximally present in the filter W . Zero mean random filters are preferred for easier optimization and extraction of wide set of features.

2.1.2 Extreme Learning Machine

ELM is a single layer feed forward Neural Network-SLFN (see Fig. 2.5). Compared to traditional back propagation learning rule that modifies all the weights, the ELM allows first layer weights W_i and bias b to be random while only the output weights, β_i needs to be tuned based on the training data T . It has been proven in [29] that SLFN with randomly generated additive or RBF nodes and with a broad type of activation functions, can universally approximate any continuous target functions. β , the output weight is calculated as least square solution given the target values of training set. The hidden layer neuron response for j -th hidden neuron h_j is given as follows.

Figure 2.5: **ELM structure with single hidden layer:** In such SLFNs implementations, the activation function $g()$ for additive nodes can be any bounded non constant piecewise continuous functions. $g : R \rightarrow R$. In this figure there are c output classes.

$$h_j = g(W_j^T X + b) \quad (\text{Eq. 2.4})$$

Output O_k of k -th class is given by $O_k = \sum_{j=1}^L \beta_{kj} h_j$, where L is number of hidden layer neurons and β_{kj} is output weight. For classification tasks the input is categorized as the k -th class if o_k is the largest output. We can define the classification output as an integer class label 's' given by

$$s = \operatorname{argmax}_k(o_k), 1 \leq k \leq C. \quad (\text{Eq. 2.5})$$

Suppose there are p training samples-then we can create a $p \times L$ hidden layer output matrix H where each row of H has the hidden layer neuron output for each training sample. Let $T_k \in R^p$ be the vector of target values for the p samples. The solution of L_2 norm minimization can be directly found as $\beta_k = H^+ T_k$, where H^+ is the Moore-Penrose generalized inverse of the matrix H . Intuitively, we can think of the first layer as creating a set of random basis functions while the second layer chooses how to combine these functions to match a desired target [4].

2.2 Literature Survey : Hardware Implementation of CNN

Earliest of the analog CMOS, hardware implementation of CNN include ANNA (Analog Neural Network ALU) chip by Bell labs [30], a mixed analog-digital processor that could compute 64 simultaneous 8×8 convolutions at a peak rate of 4G.MAC op/s. Though the chip can be programmed with 100K 6-bit weights, each MAC operation with 32 inputs consumes ~ 1 nJ. Moreover the weights need to be refreshed in each millisecond. The multiplier units occupy large area and impede vast integration of network. Digital implementation of Convolutional neural network is also limited by bit precision. Use of smaller word-length (≤ 16 bits) can lead to problems such as overflow of accumulation register, reduction in speed caused by extra clock cycle

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

needed to access double word-length multiplicands [31]. Truncation of intermediate sums and products will lead to erroneous outputs.

A programmable logic device approach for CNN Implementation: *Gokhale .et.al* presented a scalable, low-power co-processor for enabling real-time execution of deep neural networks called nn-X [3]. These included a programmable processing modules called 'collections' which perform computationally intense operations of deep CNN such as convolution, non-linearity operation etc. These collections are made of programmable logic devices.

At the system level, a host processor is included for controlling input data and programming nn-x processor with configurations for computation. 'nn-X' co-processors consist of *collections* and *memory-router* and *configuration bus*. Each *collection* consist of one convolution engine, one pooling module and one computation engine for non-linearity operation. A *collection* module caches the weights required for convolution, so that an uninterrupted pipeline can be operated. The *collection* can also smoothen the feature maps with Gaussian kernel. The non-linearity operation is implemented by piecewise linear approximation of non-linear function.

Figure 2.6: **An internal diagram of collection:** Collections are core configurable computation engines of nn-X. It performs data routing, convolutions, pooling, non-linear programmable functions. Purple lines below are configuration buses [3].

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Max-Pooling layer in *collection* has digital comparators and registers. As data is streamed into the module each pixel/feature value is compared with previous highest value stored. Data is streamed into the pooling layer row-wise into the pooling engine. A memory router connects the collections with external memory. Its purpose is to route arbitrary data streams out of collections and stream inputs into it. A separate configuration bus configures computation functions and non-linearity operations in collections (see purple bus in Fig. 2.6). Configuration bus is also responsible for multiplexing data out of memory routers into distinct collections residing in nn-X. The system successfully demonstrates simple cognitive tasks like street scene-parsing, face detection etc.

While delivering one output of 10×10 convolution layer per clock cycle, it can give a conservative through-put at a rate of 25GOPS/s. Note that 1 operation here is one addition and multiplication combined. But this consumes a power of 1W for the nn-X processor.

A custom designed analog/mixed signal CMOS solution: The convolution chip presented in [18] is an Address Event representation (AER) transceiver with an array of event based computation blocks of features. CAVIAR, "Convolution AER Vision Architecture for Real-Time " is multi-core chip for visual recognition applications. For each incoming event, integrators within a projection field around the addressed pixel compute a weighted event integration. The weight of this integration is defined by the convolution kernel. A pixel in [18] contains digitally controlled pulsing (pulsing Current DAC) current sources(positive leg) and current sinks (negative leg), which provide a current pulse of fixed width. Amplitude of the current pulse is dependent on the kernel weight stored in the dynamic register. Current pulses are integrated onto a capacitor, whose voltage is monitored by two comparators. If an upper (lower) threshold is reached, the pixel sends a positive (negative) output event. The winner take all (WTA) chip implements feature competition. WTA operation on the inputs form a feature map,

to determine the strongest input (which codes the location of the strongest feature in the feature map). This implementation requires two DAC elements per synapse. For faster outputs, the DAC setting the synapse weight has to settle very fast. This poses a big challenge for the design of DAC elements with high resolution. These require generation of complex control signals too. Also the analog /mixed signal computation faces challenges from device mismatch. These device mismatch lead to skewed computation. Note that tolerance in responses of neural network is limited to final neuron responses, not intermediate products of computation.

Simplified implementation of CNN in CMOS: The HAGEN chip implemented by *J.Fieres et. al.* [32] consists of a two stage CNN followed by a fully connected layer. The neural function implemented by convolutional layer is a simple Heaviside function. Hence, the convolutional layer outputs are explained as $O = u(\sum_i W_i I_i)$, where W_i are weights I_i are inputs to neuron.

The training method was as follows. The intermediate convolution layers were trained in unsupervised manner with a self-organized map structure. The output layer weights are trained in supervised manner with full MNIST dataset. Simple perceptron learning is used for training output layer. The pre-trained kernels are simple feature extractors like slanting edges and lines. The weights are quantized to a set of -1, 0, 1 after training. Output layer weights are also discretized to two levels. This makes hardware implementation simpler. The circuit/block diagram of one neuron, corresponding to one row of a synapse array is shown in Fig. 2.7. When inputs are weighed with switches (controlled by 'weight sign') positive and negative currents are summed into the two electric lines I_{pos} and I_{neg} . The synapses act as current sinks, connected either to the I_{pos} or the I_{neg} line, depending on the sign of the synaptic weights. The weight capacitor is charged during the programming phase. The neuron circuit converts the negative and positive leg currents into voltages, (see Fig. 2.7) which are compared in

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.7: **Basic components of HAGEN chip** : Sign bit determines, whether to sink the input current from I_{pos} or I_{neg} . The neuron is a simple comparator of negative and positive currents

order to determine neuron output.

The following implementation yields results close to 98.2% for MNIST database. The implementation provides good result for traffic sign recognition also. The power dissipation of the circuit can be minimized by operating current sinks in sub-threshold followed by a low-power current comparator. The following technique is applicable to MNIST since the digits are composed of relatively simple spatial features like lines and slants. This method cannot be scaled to bigger networks and complex database [24]. Note that the inputs are thresholded from gray scale(0-255) to black and white to produce single bit inputs. These are ineffective in larger pictures present in database such as Caltech101 etc.

2.2.1 Hardware implementation of Random Projection Core

Earlier [33] had developed a compact single transistor synapse, which can achieve up to 0.45pJ per MAC operation. The weights associated with the synapses are by virtue of fabrication mismatch, hence they are non-programmable and random. [33] aims to implement a ELM classifier for pattern classification. The implementation of random matrix multiplication in first stage (see Section: 2.1.2) can be described as

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

follow. Here, each element of input vector is converted to current with the help of a current mode DAC. The details of DAC circuits are not discussed here.

The input current (I_{DAC_m}) from each row is mirrored into all input field nodes by the current mirror array. For $0.35\mu\text{m}$ CMOS technology an 8 bit mirror resolution can be achieved with a 'C' (see Fig. 2.8) of 400fF . The ratios of the current drawn by the mirror array is essentially the input weight. These current mirrors are biased at sub-threshold regime. These weights are inherently random due to fabrication mismatch of the transistors in spite of being designed identical. The resulting weight can be deduced as follow

$$I_{m,n} = I_{DAC_m} e^{\frac{\Delta V_{T(m,n)}}{U_T}} \quad (\text{Eq. 2.6})$$

Where $I_{m,n}$ is the current drawn by the m-th row n-th column current mirror element of grid shown in Fig.2.8 and I_{DAC_m} is input current in m-th input field. U_T denotes the thermal voltage and ΔV_T denotes the threshold voltage mismatch between input transistors on m-th row and mirror transistor on n-th column of respective row.

$$W_{m,n} = e^{\frac{\Delta V_{T(m,n)}}{U_t}} \quad (\text{Eq. 2.7})$$

The weights thus obtained $W_{m,n}$ are log-normal distributed since ΔV_t follows a normal distribution. The counter output (see Fig. 2.8) of each Current controlled oscillator ADC will be sum of currents sourced by current sources in corresponding column. The computation of neuron is activated by setting NEU high. The output of CCO is a pulse frequency modulated signal with the frequency proportional to total input current (I^{in}). I^{in} is the sum of current drawn by all transistors in the column. The output frequency is counted for number of flips in the counter (see Fig. 2.8). The counter outputs are latched and serially read using the CLK OUT signal. When NEU is low, Current Controlled Oscillator (CCO) is disabled to save power. Total current I_{in} from the current mirrors starts to discharge V_{mem} , (see Fig. 2.9) until it crosses the threshold voltage of the INV_1 , leading to transition of all inverters. The transition of

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.8: **The random projection core circuit:** Inputs are fed to current array with help of current DAC elements [4]. The input current gets mirrored in all grid elements. All current mirrors in a column draw total current I_{in} from one Silicon neuron. Thus the desired MAC operation response is generated in respective counters. CLK_{OUT} is used for shifting counter outputs out in 13 bit bus. The implementation in [4] consisted of 128 input channels preferably enabled with address line $A < 0-7 >$.

INV_2 is coupled instantly to V_{mem} through C_f pulling C_{neu} down. At the same time, M_3 turns on, charging V_{mem} towards V_{DD} until it crosses the threshold voltage of INV_1 from low to high and the cycle repeats. Output is obtained from INV_3 and registered in a counter. The cycle time of CCO operation is the sum of charging time T_1 and discharge time T_2 of C_{neu} . Thus the total cycle time T_{cycle} , is $T_1 + T_2$ given by :

$$T_{cycle} = C_{neu} \cdot V_{DD} \left(\frac{1}{I_{in}} + \frac{1}{I_{rst} - I_{in}} \right) \quad (\text{Eq. 2.8})$$

$$f_{cycle} = \frac{I_{in}(I_{rst} - I_{in})}{I_{rst} \cdot C_{neu} \cdot V_{DD}} \quad (\text{Eq. 2.9})$$

f_{cycle} is the output of counted in counter. The counter output for increasing I^{in} can be plotted as shown in Fig. 2.9 (b). When $I^{in} \ll I_{rst}/2$, we can obtain an almost linear relationship: $f_{cycle} = \frac{I^{in}}{C_{neu} \cdot V_{DD}}$. These counter outputs are essentially the hidden layer neuron responses of ELM in [4]. The digital multiplier blocks were used in [4] for

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.9: **The neuron used for pulsed output generation:** a) schematic of the current mode neuron circuit used in [4] b) The counter output of the CCO-ADC varies according to dotted line shown. Neuron spiking frequency initially increases with the increase of the input current I^{in} , until $I^{in} = I^{rst}/2$. It then reduces and becomes zero finally when $I^{in} = I^{rst}$.

implementing output layer.

2.3 A Low Power Hardware Implementation of CNN

This section discuss a new low power feature extraction on backgrounds explained in section 2.1. The structure and circuit shown in section 2.2.1 can be used to implement convolution layers of a CNN. Convolution layers by virtue of computations involved consume most fraction of power and area in a fully digital implementation of CNN. By re-arranging the image patch into one input vector, each computation (output of CCO-neuron) provides one pixel output of N^1 feature maps. The weights of i -th column in current-mirror array become weights associated with W_i^{th} convolutional kernel. This technique which use the same random projection core as convolution kernel is called " RP^1CNN ". The performance of such a feature extraction technique is limited. The detailed analysis of [28] points out the requirement of a zero mean random distribution for feature extraction with CNN networks.

A second technique " RP^2CNN " is thus proposed for extraction of features. Here, the difference in responses of two adjacent counter outputs is considered as one pixel

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.10: **Proposed solution RP²CNN** . Input patch under consideration is rearranged as a vector and are shifted to input registers. Output responses of two consecutive CCO-ADC are subtracted to form one pixel output in output feature map. A cyclic pairwise subtraction can earn FM¹ feature maps. By scanning the input image once we can produce FM¹ feature maps.

in extracted feature map. For eg. column 1 (red)- column 2 (blue) (in Fig. 2.10) gives:

$$W'_1 = [w_{11} - w_{12} \quad w_{21} - w_{22} \quad \dots \quad w_{l^2_{21}} - w_{l^2_{22}}], \text{ forms the new } W'_1. \quad (\text{Eq. 2.10})$$

Figure 2.11: **Processing stages were Random projection core can be used.** An efficient pipeline of the deep network is proposed as shown. Pooling layer outputs are generated with simple digital shift registers and adders. The first stage of ELM classifier can be implemented with the same random projection technique. Finally the output classes are obtained from preloaded output weight β .

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

x_1	w_{11}	w_{12}	$\dots$	w_{1FM}	x_2	w_{11}	w_{12}	$\dots$	w_{1FM}
x_2	w_{21}	w_{22}	$\dots$	w_{2FM}	x_3	w_{21}	w_{22}	$\dots$	w_{2FM}
$\vdots$					$\vdots$				
x_i			w_{ii}		x_{i+1}			w_{ii}	
$\vdots$					$\vdots$				
x_{ℓ^2}	$w_{\ell^2 1}$	$w_{\ell^2 2}$	$\dots$	$w_{\ell^2 FM}$	x_1	$w_{\ell^2 1}$	$w_{\ell^2 2}$	$\dots$	$w_{\ell^2 FM}$

Figure 2.12: **Rotation of inputs field:** Rotation of bits results in each input field being multiplied by new random vector, hence producing distinct feature map.

By column-wise subtraction of adjacent vectors in a cyclic way we can generate FM^1 pixel outputs of all FM^1 feature maps. This implementation of Y_{ij} as difference of two CCO-DAC outputs, we can yield useful first layer outputs. The index of input vector field is specified by $A < 0 : 6 >$ address line. Table 2.2 presents improvement of results for " RP^2CNN " over " RP^1CNN " approach.

Requirements of second stage of CNN: The same random projection core can be reused for producing different set of random filters for second stage filters. The number of second stage random filters required are quite high for all databases. (see Table 2.1) This can be achieved by rotating the inputs and outputs. Rotation of inputs by one address bit produces different random filter for our application. Each rotation of input vector index produces FM^1 new feature maps. It is shown by [33] that with a current mirror array of size $P \times Q$, $(P \times Q)$ random weight vectors of can be emulated (see Fig. 2.12). By integrating the core random projection circuit with other basic digital buffers and arithmetic units and we can form an end to end system for classification task.

Peripheral circuitry for end to end classification: A digital buffer and output multiplexer are necessary for providing, input patches of output feature maps to pooling layer. All the compressed feature maps from pooling (second) layer are not required for computing all third layer features. The buffer size required is optimized according to number of instances each second layer feature map needs to fanned out. The extracted features are rearranged and fed to an independent ELM classifier. Since ELM

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.13: **Architecture of complete image recognition system.** All digital processing stages can be implemented inside a DSP peripheral block. The inputs of random-projection core are arbitrated for maximum utilization. The connection matrices of CNN stages are stored in routing stages for optimized for maximum throughput of pipeline.

first layer also involves random projections, the whole network can be implemented with the help of an architecture shown in Fig. 2.13. 16 random projection cores can be used for implementing Convolution stages and first stage of ELM classifier. With the help of fast routing engines and digital blocks, a pipeline of steps shown in Fig. 2.11 can be setup.

2.4 Results

The responses from a current mirror array fabricated in a $0.35\mu\text{m}$ CMOS technology were used for characterizing Random Projection core. The weights in the current mirror array were obtained by writing input values into each input row and obtaining CCO responses from all columns. The weight corresponding to RP^2CNN were obtained by cyclic difference of CCO output responses. The test-setup used to characterize the weights of the current mirror array is shown in Fig. 2.14(a). The weights obtained, which are the ratio of input current to CCO output response follow Log-

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.14: **Test setup used for characterizing weights of current mirror array:** a) Inputs to each row is transferred into the shift register through a serial link from a 'Spartan 3' FPGA. After obtaining the output CCO responses, they are shifted into the FPGA and transferred to PC. b) Die photograph of the current mirror array fabricated in $0.35\mu\text{m}$ CMOS technology.

normal distribution (see Fig. 2.15(a)). The measurements show a standard deviation (σ) of 16mV for the Gaussian distribution of ΔV_T .

The random weight distribution obtained through the proposed method RP^2CNN gives symmetric distribution of weights around zero. These weights obtained were used to simulate Convolutional neural network (CNN) with design parameters from Table 2.1. For fast and efficient processing of the pipeline following modifications were imparted to the network without harming the CNN efficiency.

Details of parameters tuned in CNN: The outputs of feature extraction layer were quantized in the simulation to 8 bits resolution. Number of second stage filters were optimized without reducing the orthogonality of extracted features. For fast and simple digital processing of Sub-sampling and LCN stage, divisors and multiplicands are approximated to exponentials of two. Careful approximation of these co-efficients

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.15: **Distribution of weights in the chip:** a) Figure shows distribution of Log-normal weights observed by virtue of mismatch in the chip. b) A zero-mean distribution of weights obtained for feature detection by subtracting adjacent column counter outputs.

lead to negligible loss of performance. The coefficients of Gaussian filter for Local contrast normalizations are made powers of two, which enables faster calculation of normalized features by shifting of multiplicands.

MNIST hand written digit database consist of 60,000 training samples and 10,000 test samples. Table 2.2 shows advantage of RP^2CNN for MNIST database. The first layer features obtained through " **RP^2CNN** " were capable of extracting simple features like edges and curves of input images (see Fig. 2.16). The features also gave

Table 2.2: Comparison of two random feature extraction methods.

# of MNIST Training samples	RP^1CNN testing accuracy(%)	RP^2CNN testing accuracy (%)
300	86.0	91.5
1000	88.0	94.5
2000	91.0	96.0
6000	92.0	98.0

Table 2.3: Testing accuracy of the network in Matlab simulation.

Dataset	"RP ² CNN" (%)	Prior work results (%)
Caltech101	63.4	64.3 [25]
MNIST	99.49	99.61 [2]

maximal response for distinct features of input image. This points to specificity of generated feature maps and spatial invariance of features in final feature maps.

We further classified the entire MNIST database and Caltech 101 database. Caltech 101 [27] is a database of natural images of size varying from 140 x 140 to 220 x 300. We used 30 training samples and 5 testing samples from each 100 class of images from Caltech101 database. Table 2.3 shows result obtained for different database tested with proposed RP²CNN feature extraction method. Prior work results are accuracies obtained from references, which reported software results for random weight based feature extraction.

CHAPTER 2. HARDWARE ACCELERATION OF ARTIFICIAL NEURAL NETWORK: A CMOS SOLUTION

Figure 2.16: **First layer feature maps:** Figure contains first layer features maps of inputs a) Digit eight. b) butterfly. and c) camera. Different feature responses are seen high in different maps.

Chapter 3

Post-CMOS Solution for Self-Learning Low-power Synapse

A big challenge in the field of neuromorphic systems that aim to implement low-power, brain-inspired integrated circuits to impart cognition to robots interacting with environment in real-time is the design of a self-learning synapse [34]. Since synapses outnumber neurons by a factor of 100X, a synapse needs to be much-more compact and power-efficient than a neuron. We know that with increasing speed of computation in deep nanometer CMOS technologies, memory operations are not able to increase their speed in same pace. This forces us to choose a embedded memory based synaptic implementation. Here we focus on neuromorphic circuits which implement SNN with self-learning synapses. The goal of this chapter is **a) feasibility of Domain Wall Magnet based self-learning synapse is shown with co-simulation of circuit blocks and Spin-transport modules. b) The low-energy, low-voltage requirements for write operation of synapse is highlighted. c) Online learning capability of SNN implemented is shown.**

3.1 Background: Spiking Neural Network (SNN) and Spike Timing Dependent Plasticity (STDP)

With recent advances in neurosciences, we have been able to accurately model effect activities between neurons on synaptic strength. *W.mass* in his theoretical work [35] have shown that spiking neurons with temporal coding have more computational power than sigmoidal neurons. These findings have opened up many possibilities for event driven neural network. The neurons in initial models of spiking neural networks employ a rate based encoding of output signals where higher frequencies of output spikes encode a higher output signal. The output frequencies are calculated in windows of time. This produces an average response for the inputs provided to the neuron. The adjustment of synaptic strengths will most likely alter the neuron's output firing rate. This simple basis for learning is known as synaptic plasticity. Simple gradient-descent method can be used for learning the synaptic strength, if we use continuous activation in neurons [36]. Later generation of SNN are more close to biological neural network. Here the neurons use pulse coding, by virtue of which they can receive and send information as discrete spikes. These allow communication and computation in the SNN as instantaneous time and amplitude of spikes. There are many dynamic models of neurons which produce asynchronous spikes in literature like Leaky integrate and fire and Quadratic integrate and fire etc. When SNN is realized on hardware, the circuit is activated only on sparsely spikes or events need to be communicated between neurons. This will minimize dynamic power consumption.

Spike Timing Dependent Plasticity (STDP) is one of the biologically plausible learning rules commonly used for modifying synaptic strengths in SNN [5]. In the STDP learning rule, pre-synaptic spike arrival before post-synaptic action potentials leads to long-term potentiation (LTP) of the synapses, i.e. weight change ΔW a function of the time difference, Δt between pre- and post- synaptic spikes as shown on the right side of Fig. 3.1. On the other hand, pre-synaptic spike arrival after post-synaptic

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

spikes leads to long-term depression (LTD) of the same synapse.

$$\Delta w = \begin{cases} \Delta w^+ = A^+ e^{(\frac{-\Delta t}{\tau_+})} & \text{if } \Delta t \geq 0 \\ \Delta w^- = -A^- e^{(\frac{\Delta t}{\tau_-})} & \text{if } \Delta t < 0 \end{cases} \quad (\text{Eq. 3.1})$$

where $\Delta t = t_{\text{post}} - t_{\text{pre}}$ is the time difference between a post-synaptic spike and pre-synaptic spike, τ_+ and τ_- are the time constants of the learning window, and A^+ and A^- are the maximal weight changes for potentiation and depression, respectively. In hardware realization of STDP memory traces are widely used to encode the time since last pre/post-synaptic spike. These are mostly encoded in amplitudes of waveforms.

Figure 3.1: **STDP characteristics from [5] is shown.** t_{pre} and t_{post} corresponds to time instances of spike events of pre-synaptic neuron and post-synaptic neuron respectively.

In STDP, synapses compete for control of the timing of post-synaptic action potentials. Synapses that fire post-synaptic neuron immediately upon arrival of pre-synaptic spike with or synapses that are connected to inputs that act in correlated groups will be able to compete most successfully and improve their synaptic strength, while synapses which produce response slow or have higher latencies to inputs are weakened.

3.2 Literature Survey of Self-Learning Synapses for SNN

3.2.1 Self-learning Synapse in CMOS

The neuromorphic circuit implemented by *Giacomo et.al* [6] in $0.8\mu\text{m}$ CMOS process consist of a 2-D array of 32×6 self-learning synaptic circuits. The chip consist of 32 integrate and fire (I&F) neurons with each connected to 6 excitatory synapses. The output of all 32 neurons can be communicated to arbitrary number of input synapses or interfaced with off-chip test circuitry. Each synapse in the network consist of four parts namely STDP circuit, short term depression STD circuit, Bi-stability circuit and current mirror integrator CMI circuit. The synaptic weight is stored as V_w in the synapse. Input spikes denoted by 'pre' extracts charge from CMIs integrating capacitor according to voltage V_w . The input current I_{inj} to integrating capacitor of I&F thus set by V_w .

The STD circuit is used to implement local gain control of injected current I_{inj} ,

Figure 3.2: **Synapse circuit from [6]:** The CMI circuit implements a current-mirror-integrator and generates a post-synaptic current that is injected into the neuron. The STD circuit necessarily acts as a high pass filter of inputs spikes weighing the initial stimulus more than subsequent input activity. The STDP based learning is implemented in the synapse with the help of memory traces V_{pot} and V_{dep} . This circuit, that drives the synaptic weight to one of two possible states on long time scales.

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

for stimulus specific adaptation. These are computationally useful according to [37]. In absence of stimulation V_w is equal to V_{w0} . Through the course of input spikes V_w is gradually reduced at a rate set by V_w . For the implementation of STDP learning curve in CMOS synapse, a waveform V_{pot} is generated on arrival of pre-synaptic spike and a waveform V_{dep} is generated on arrival of post-synaptic spike. The pre-/post-synaptic spikes also generate pulses on gate of transistors M_8 and M_5 respectively. The sampling of V_{pot} by post pulse allow a current I_{pot} to flow into C_w increasing weight. The sampling of V_{dep} by pre- pulse allow a current I_{dep} to sink from C_w decreasing weight.

Conventional VLSI capacitors are not ideal in that they slowly leak the charge they are supposed to store. Bistability circuit slowly drives the synapse weight voltage V_w to high or low after comparison with a threshold voltage. Moreover the circuit have a integrate and fire (I&F) neuron, for which an explicit threshold can be set. The I&F neuron also have spike frequency adaptation, which helps in reducing bandwidth required for communication of spike events. The circuit implements STDP characteristics successfully. The circuit, though can store only two states of weight in long term.

3.2.2 Floating Gate based Self-learning Synapses

Floating Gate devices are one of the non-volatile memory device employed for implementing self-learning synapse. The floating gate transistors are capable of storing non-volatile weights in the form of floating gate voltage, V_{fg} . When pre-synaptic inputs are provided as V_g (see Fig. 3.3), the drain current of a sub-threshold saturated FG device is given by: $I_d = I_0 e^{\frac{\kappa(V_{dd} - V_{fg})}{U_T}}$, where U_T is the thermal voltage and κ is the gate coupling coefficient. Rewriting drain current, we get:

$$I_d \propto W e^{(\Delta V_g)} \Rightarrow W = e^{\kappa V_{fg}/U_T}. \quad (\text{Eq. 3.2})$$

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

Figure 3.3: **Floating gate transistor and terminal waveforms** : a) The FG transistor's terminals with corresponding waveforms for LTD and LTP [7]. b) The system configuration adopted for Program mode.

Weight modification in a FG synapse uses a combination of hot-electron injection and Fowler-Nordheim tunneling [38]. Hot-electron injection adds electrons on to the floating gate node, which reduces the floating gate voltage resulting in more current through the transistor hence increasing the weight of the synapse. On the other hand tunneling removes electrons from the FG node to reduce the synaptic weight. For implementation of STDP learning curve in FG synapse the following voltage waveforms are employed. At every pre-synaptic spike, a triangular gate voltage waveform is generated while at every post-synaptic spike, a triangular tunneling voltage and a drain voltage pulses are generated. This describes the mechanisms for programming weight in FG devices.

The work [7] describes the methodology to integrate self-learning with in fully connected neuronal circuit. When a post-synaptic spike event occurs, all the synapses connected to the neuron is turned into 'PROGRAM' mode, and the inject and tunnel control blocks are activated (see Fig. 3.3). When an input event occurs (pre-synaptic spike), the pre-synaptic computation block produces a triangular waveform which feeds into all synapses connected to input. Though these circuitry enable STDP

learning in circuit, The voltage required in injection and tunneling are high. These large voltages degrade reliability of underlying CMOS circuitry. As described in appendix B, the energy required for implementation of learning algorithm is also high.

3.2.3 Memristor based Self-learning Synapse

Memristors are other suitable candidates for a synapse due to their non-volatile nature. Memristors are two terminal devices which change their conductance when charge or flux flow through them. These conductances act as weights when relatively low read out voltage are applied across the terminals. They can be programmed to distinct resistance levels with voltage pulses. Typically, resistance switching devices creates localized conducting paths (filaments) whose motion results in discrete, abrupt resistance switching characteristics [39]. The fabrication process (see details in [39]) followed by *S H Jo et.al.* is able to perform analog switching behaviors. The width of the pulse applied across the terminals of memristor synapse decides the weight percentage change, i.e conductance change here. The polarity of voltage applied will determine whether the weight would increase or decrease.

The outputs of neurons are evaluated only on dedicated time slots. In [39] the spikes produces synchronized pulses of different width in pre-determined time slots, as response to evaluation. The memory trace or time since last spike is encoded as decaying pulse widths. These pulses have half the amplitude of threshold needed for initiating conductance change. On correlated occurrence of post synaptic spike after post-synaptic spike weight potentiation happens. The synapse can show desired STDP characteristics. The voltage required across the terminal of memristor, for LTP and LTD are 3.3 and -2.9 V respectively. Application of such high voltage would reduce the reliability of CMOS substrate. Moreover, these memristor based synapses consume high energy for modifying associated weight.

3.3 Enabling STDP with DW based Self-learning Synapse for SNN

Last decade saw increased interest in spintronics devices and spin transfer torque -random access memory (STT-RAM). These STT-RAM devices are scalable ,consumes very low power in write operations and are extremely scalable. Their endurance levels are high compared to other non-volatile counterparts like flash [40]. The recent advances in theory of spin based interaction with ferromagnetic materials and material science research have made Stuart Parkin's racetrack memory [41], which is a solid state memory unlike other Magnetic Random access Memory (MRAM) realizable today. These features make spin based devices a possible candidate for a system with in memory computation.

3.3.1 Introduction to Spin based devices

A charge current is the flow of electrons or holes. These electron also posses spin. The flow of these spins leads to spin current. In non-ferromagnetic materials, the electrons spins have random orientations which lead to zero net spin current. When these electrons currents pass through ferromagnetic materials the experience a transfer of spin angular momentum. This gives rise to a spin polarized current. Spin polarized current is of significant importance both scientifically and technologically. As mentioned earlier, a charge current entering the magnetic material gets spin polarized i.e. a fraction of conduction electrons' spin align with that of the magnet. The ability of the magnet to polarize a charge current is quantified by the polarization factor P . Conventionally, the magnetization of a material or device is switched by applying external magnetic fields. Using spin transfer torque phenomena the same could be achieved in nano-structures by injecting spin-polarized current. With this spin-transfer torque effect it is possible to switch the magnetization of a nano-magnet.

Figure 3.4: **Lateral spin valve** : The vertically inserted input charge current generates a spin polarized current in the channel. The charge current in the channel flows into the large ground lead whereas the spin polarized current is absorbed by the small output magnet. Ground lead can be biased at a voltage close to power-supply for providing adequate input current. Hence 'non-local spin transfer torque' was exploited several ultra low-power boolean and non-boolean applications [8].

Non-local spin transfer torque Fig. 3.4 shows a Lateral Spin Valve (LSV) structure. A charge current flowing to ground through a magnetic contact on the left (input nano-magnet), gives rise to a spin current to the right, outside the path of charge current. The output magnet's magnetization is reversed due to the non-equilibrium spin-accumulation under it. This non-local spin current was first observed in experiments [42] of *Y.Otani* and colleagues in 2008. Later these results led to the proposal of All Spin Logic (ASL) circuits by *S.Datta et.al*. Through careful benchmarking against available theory and experiment ASL established a set of elemental modules representing diverse materials and phenomena. These elemental modules can be integrated seamlessly to model composite devices involving both spintronics and nano-magnetic phenomena. These modules are explained in Appendix A.

3.3.2 Spin based Neuron module for Non-Boolean Computation

A domain wall strip can be explained as two oppositely polarized magnetic domains separated by a Domain wall (DW). The domain wall thickness depends on the anisotropy of the materials. It is shown in several studies [23], [43] that DW motion can

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

Figure 3.5: **DW magnet strip as synapse** a) Position of DW magnet w.r.t center of the strip becomes of weight W of the DW-synapse. The symbol shown in right side will be used for further representation. b) Spin injection strength into channel for different charge current X_{in} as a function of W [9].

be achieved with very low current densities for soft magnetic materials with high Perpendicular Magnetic Anisotropy (PMA). These DW Magnet structures can be used to efficiently implement non-boolean computations of an artificial neural network (ANN) as shown next. The output of a simple neuron is given by

$$Y = G\left(\sum_i W_i X_i + b\right) \quad (\text{Eq. 3.3})$$

where W_i is weight of i^{th} synapse, X_i is input to i^{th} synapse, b is the bias of the neuron and $G(\cdot)$ is the activation function. It has been shown in [9] that spin-current injection strength into the channel depends on position of the domain wall from the center of domain wall strip. Thus the position of domain wall from the center gives the bipolar weight term W . As proposed in [9] input currents inserted vertically to the DW Magnet enter the channel after getting polarized (see Fig 3.5). Fig. 3.5(b) shows the variation of the weighted spin currents (WSC) according to the position of the domain wall when two different magnitudes of input current ($3 \mu\text{A}$ and $6 \mu\text{A}$) are applied. These WSC gets summed up in the channel to produce I_{net}^{spin} . As shown in Fig. 3.6(b) the net spin current is absorbed by an output magnet. This net spin current exerts a spin torque on

the output nanomagnet. By employing Bennet clocking scheme, the net spin current required to reverse the output free layer magnet direction, can be reduced [44].

A preset magnet is added to the structure (see Fig. 3.6(b)) which has its magnetization directed orthogonal to the output magnet. On injection of a current pulse through the preset magnet, the output magnet aligns itself to the hard axis. The final state of the output magnet is decided by the net spin current present (I_{net}^{spin}) in the channel on release of preset pulse (current pulse through the preset magnet). The output magnet forms the free-layer of a (Magnetic tunnel Junction) MTJ stack. The MTJ output is read using a differential latch and thus converted to electrical signal [9]. Transferring spin signals over large distance is very difficult, hence the neuron outputs are transmitted to other neurons as an electrical voltage. This Bennet clocking scheme reduces the threshold required to flip the output magnet to limits set by noise currents. The above procedure also makes the output evaluation tolerant to variations in neuron magnet parameters. Thus the MTJ output can be represented by:

$$Y = \text{sgn}\left(\sum_i W_i X_i(t) + b\right) \quad (\text{Eq. 3.4})$$

where $X_i(t)$ are inputs and W_i are weights as dictated by DW position. Note the similarity between this equation and equation (1) –this circuit implementation uses the sign function as the neuron's activation $G(\cdot)$ used in equation (1).

3.3.3 DW Motion Characteristics

It is known that DW motion can be controlled by varying both amplitude and duration of current pulses. The Domain wall strip proposed here has constrictions or notches at desired positions. These constrictions act as local pinning sites which prevent motion of Domain wall due to thermal fluctuation [45]. Fig. 3.7(a) shows the final position of displaced DW, along a strip for different pulse amplitudes. Here we use one-dimensional model of Domain Wall Motion along a nano-strip according to [46].

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

Figure 3.6: **Sequence of events in MTJ output evaluation**; [9]. a) A preset pulse pushes the free-layer magnet orientation (M_z) to meta stable state. As soon as the preset pulse is removed (I_{net}^{spin}) decides the orientation of free-layer magnet. Clock pulse is synchronized with the preset pulse for MTJ o/p shown in Fig.3.12(b). b) A preset magnet is added in the structure along with DW magnets shown in Fig. 3.5. The magnetization of preset magnet is directed orthogonal to output free-layer magnet [9].

The simulation setup is discussed in appendix. The one-dimensional model described in (Eq. 3.5) and (Eq. 3.6), saves cumbersome mathematical simulations and time. By providing good estimates of anisotropy field H_k and Domain wall width Δ , better insights and trends can be observed. Domain walls, in one-dimensional approach, are seen like dots moving along a line.

$$(1 + \alpha^2) \frac{\dot{X}}{\Delta} = \alpha \gamma_0 (H_e + H_p(X) + H_{th}(t)) + \frac{1}{2} \gamma_0 H_k \sin(2\Phi) - (1 + \alpha\beta) \frac{b_J}{\Delta} \quad (\text{Eq. 3.5})$$

$$(1 + \alpha^2) \dot{\Phi} = \alpha (H_e + H_p(X) + H_{th}(t)) - \frac{1}{2} \alpha \gamma_0 H_k \sin(2\Phi) - (\alpha - \beta) \frac{b_J}{\Delta} \quad (\text{Eq. 3.6})$$

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

Figure 3.7: **Displacement achieved with different current pulse amplitudes** a) Width of pulse used here is 100ns b) Final position of Domain wall normalized w.r.t notch interval, on application of current pulse with amplitude shown in x-axis. Maximum and minimum limits of weight W will be 8 and -8 respectively, limited by the number of notches included.

where $X = X(t)$ is the position of the Domain walls center, $\Phi = \Phi(t)$ is the tilt of the Domain wall magnetization, β non-adiabatic constant, α is damping parameter, $H_p(X) = -\frac{1}{2\mu_0 M_s L_y L_z} \frac{\partial V_{pin}}{\partial X}$ in vicinity of notch, Δ is the Domain wall width, V_{pin} is the abstract measure of notch strength, and $b_J = \frac{J_a \mu_B P}{e M_s}$ is the term from current density J_a and polarization P of ferro-magnet. The experiments considered here are limited to 0K and hence the stochastic thermal field term H_{th} is zero.

The model is benchmarked with [46]. A thin permalloy strip of cross section (60nm $\times$ 4nm) is considered. The parameters used for simulation (see Table 3.1) are in close agreement with [43].

Periodic triangular notches (8nm wide) at an interval of 80nm, acts as local pinning site in the nano-strip. V_{pin} , the peak pinning potential which is an abstract measure of pinning strength is assumed as 10^{-20} J. The quantity is of magneto static nature [43]. The results of [43] reveal that an initially pinned domain wall can be expelled away from the constriction with a current pulse of sufficient amplitude. But if the maximum displacement does not surpass a given restrain-length (L'_n), the domain-wall experiences an attractive force $H_p(X)$, which pulls it again towards the initial pinning site, on release of current pulse. This restrain length (L'_n) observed in study is much more than notch width [43]. By increasing the amplitude of current pulses further, more constrict-

Table 3.1: Magnetic parameters of DW strip.

Parameter	value
Exchange Energy Constant (A)	$1 * 10^{-11} J/m$
Damping parameter (α)	0.01-0.1
Saturation Magnetization.-(M_s)	860kA/m
Nano strip dimensions (Length /Breadth/thickness)	$1.2\mu m \times 60nm \times 4nm$
Non-adiabatic Parameter (β)	0.0-0.2
Uniaxial Anisotropy-(H_k)	2000-3000 A/m

tions can be surpassed. The DW also moves backwards, on application of negative current pulse along the strip. By carefully choosing notch interval, the following steady state response is obtained for a typical value of parameters. Figure 3.7(b) shows weight change ΔW , quantized in terms of interval of notches, as a function of amplitude of J_a . For learning the synaptic weights i.e. W, DW position have to programmed with corresponding write currents as shown in Fig. 3.7(b).

An undesired effect in spin-based neuromorphic circuit is shown in Fig. 3.8 namely, cross talk. When an input charge current I_r injected in an input DW magnet, a small component of spin current reaches adjacent DW magnet through the non-magnetic channel. An oxide interface like MgO is required in the between input magnet and channel to increase the transmission efficiency of spin signals. It is seen that the number of conducting states in the input-magnet is several orders larger than the conducting states in the non-magnetic channel. As a result, in spite of having a large spin polarization factor, p for input magnet, the spin current injected into the semi conducting channel enters the completely filled states in the channel. This reduces the spin injection efficiency of the structure. A thin oxide layer suppress the overall trans-

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

mission of spin signals and hence producing significant difference in spin currents of opposite polarities in the channel.

Fig. 3.9 shows the effect of interface oxide thickness t_{ox} , on crosstalk current I_{CT} . Larger t_{ox} increases the interface resistance and hence suppresses the absorption of spin current from the channel. However, reducing I_{CT} does minimize the wastage of spin current that is meant to be injected into the neuron. Hence, thicker t_{ox} at the DWM interface can improve the overall spin-injection efficiency of the neuron. Beyond a certain limit however, increasing the t_{ox} , increases the interface resistance beyond the DTCS transistor (M_1 in Fig. 3.10) supplying input current (I_r) to the DWM synapses, thereby mandating larger voltage drop, for the input current. Larger channel thickness T_{ch} further reduces the I_{CT} , as it lowers the effective channel Spin resistance between input DW Magnet and the neuron magnet. In case direct (local) injection is used for the neuron, larger T_{ch} can be employed to minimize I_{CT} . However, in case of non-local injection larger thickness of channel and ground lead (below the neuron) will lead to proportional increase in loss of spin current into the ground Lead, thereby rendering this method ineffective.

For a 2nm thick DW magnet the critical I_{CT} is found to be $2\text{MA}/\text{cm}^2$ which is $160\mu\text{A}$

Figure 3.8: **Cross talk current I_{CT}** : The stray spin current from one DW magnet may enter another adjacent DW magnet and disturb the domain wall position and hence weight too.

for a $256\text{nm} \times 32\text{nm}$ DW magnet. Critical I_{CT} is defined as the minimum value of I_{CT} that can cause significant change in the average magnetization of DWM2. The results show that for thickness of 4nm, The critical I_{CT} for with cross section $256\text{nm} \times 32\text{nm}$ DWM is $280\mu\text{A}$ which is order of magnitude higher than I_{CT} for even a 1nm thick oxide. ($15\mu\text{A}$ for $30\mu\text{A}$ input current). Hence, disturbance of DWM state due to I_{CT} may not be a serious concern. However, reducing I_{CT} does minimize the wastage of spin current that is meant to be injected into the neuron. Hence, thicker t_{ox} at the DWM interface can improve the overall spin-injection efficiency of the neuron. Beyond a certain limit however, increasing the t_{ox} , increases the interface resistance beyond the DTCS transistors supplying currents to the DWM synapses, thereby mandating larger voltage drop, for the input current.

3.3.4 Enabling STDP with DW synapse

For generating these write currents as a function of Δt , a circuit akin to [48] was used. The spike waveforms of the pre- and post-synaptic neurons are denoted by S_{pre}

Figure 3.9: **decrease in I_{CT} with increasing channel thickness:** increasing t_{ox} increases interface resistance. T_{ch} is thickness of non-magnetic channel.

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

and S_{post} , respectively. The time since last spike of the pre- and post- synaptic neurons can be encoded in two waveforms, T_{pre} and T_{post} , respectively (see Fig. 3.10(b)). These exponentially or linearly decaying waveform can be generated by a discharging capacitor through a resistor or a current source, respectively. The magnitude of the waveform encodes the time since the last event or spike, hence the name memory trace.

Fig. 3.10(a) shows the generation of write current by sampling $T_{pre}(T_{post})$ by $S_{post}(S_{pre})$. When a post-synaptic spike S_{post} happens at t_1 after the pre-synaptic spike, S_{post} samples the T_{pre} waveform at instant t_1 to generate a write current. Similarly, when a pre-synaptic spike happens at t_2 after the post-synaptic spike, S_{pre} samples the T_{post} waveform at time instant t_2 to generate a negative write current. So the STDP characteristics can be changed according to the memory trace. The STDP characteristics will be expressed in terms of $\epsilon = T_{pre}/T_{post}$ henceforth.

Figure 3.10: Write circuitry of DW Magnet synapse a) Write current generation from sampling of $T_{pre(post)}$ by $S_{post(pre)}$. At instance t_1 , a post synaptic spike S_{post} happens which samples T_{pre} waveform. The resulting write current creates a positive ΔW according to Δt . On the other hand, a negative current is generated by a pre-synaptic spike S_{pre} , at time instance t_2 , to decrease the weight 'W' of the corresponding synapse. b) Generation of $T_{pre/post}$ according to the memory trace. c) PMOS stack (M_3 - M_6) used to generate the write current to obtain the desired ΔW . V_0 and Δv are 750mV and 50mV respectively. d) Compact representation of the whole DW magnet based synapse completed with the write circuit. The deep triode current source transistor M_1 provides input charge current I^r .

The PMOS transistor stack (Fig. 3.10(c)) was employed to generate the write currents I^w for obtaining the STDP characteristics shown in Fig. 3.13(a). The transistors, $M_3 - M_6$, are sized such that a $\Delta V = 50\text{mV}$ with $V_0 = 0.75\text{V}$ is sufficient to produce the maximum write current I_{max} for ΔW_{max} in Fig. 3.13(a). The same T_{pre} signal is converted into an input read current I^r to the synapse through a transistor M_1 operating in the deep triode region. The read and write operations are coupled by an active low signal generated from the NOR-operation of S_{post} and S_{pre} . As a result, transistor M_2 is turned off when the DW is moved by the write current. This decouples the learning operation, which is localized at a $S_{pre/post}$ pulse, from the regular operation (Section 3.3.2), which is guided by the read currents that span a longer time duration governed by $T_{pre/post}$.

3.3.5 System Integration

As shown in Fig. 3.11(a), input charge currents $I^r(t)$ are injected vertically into the DW magnets to produce the corresponding WSC in the channel. These WSC form a net spin current (I_{net}^{spin}) in the channel which is non-locally absorbed by the output magnet. The steps involved in evaluation of final MTJ state and role of preset pulse was explained earlier in section 3.3.2. The change in resistance of the MTJ stack can be sensed by comparing it with the resistance of a reference MTJ in a differential latch (see Fig 3.11(a)) to produce a binary output [9]. Thus the MTJ output Y connected to k DW synapses can be expressed as:

$$Y(t) = u(I_{net}^{spin}) = u\left(\sum_i^k W_i I_i^r(t) + b\right) \quad (\text{Eq. 3.7})$$

where $u(\cdot)$ denotes a Heaviside function, $I_i^r(t)$ is the input current to the i -th DW magnet and W_i is its weight. Here +z magnetized output magnet produces $Y = 1$. This equation is similar to (Eq. 3.3) where the Heaviside function represents the neuronal nonlinearity.

The MTJ output is evaluated by a clocked latch and hence becomes a synchronous signal. Since each perceptron only gives a 1-bit signal, a set of p parallel

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

perceptrons is used to boost its classification power. This structure is also analogous to the biological dendritic branches [49]. Denoting the clock pulses by $p(t)$, the synchronous branch current $I_b(t)$ (see Fig. 3.11(a)) for any dendritic branch can be written as

$$I_b(t) = Y \times p(t) \quad (\text{Eq. 3.8})$$

Many such currents ($I_b(t)$) are combined to give the total input current $I_{in}(t)$ to the Integrate and Fire (I&F) neuron. The evolution of membrane potential V_m of the I&F

Figure 3.11: **System integration** a) Clocked differential latch employed to detect MTJ output change. b) Signal flow in neuromorphic circuit: Inputs $X_{in}(t)$ are propagated through the neuromorphic circuit to produce synchronous current pulses $I_{in}(t)$. This $I_{in}(t)$ is integrated by I&F neuron to produce asynchronous spikes.

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

Figure 3.12: **Integrate and fire Neuron and membrane potential evolution** a)Time evolution of membrane potential V_m (from SPICE simulation) as a result of clocked MTJ output evaluation (see Fig. 3.11(a)). The ACK signal resumes the integration of total input current $I_{in}(t)$ by I&F neuron. b)Integrate and fire neuron from [10]: $I_{in}(t)$ is the total input current to I&F neuron. The leak current of the I&F neuron can be tuned by potential V_{LK} . Hence V_{LK} and capacitor C will decide τ_m in equation (Eq. 3.9). ACK and REQ are handshake signals to facilitate large number of connections

neuron can be expressed as:

$$\tau_m \frac{d(v_m)}{dt} + v_m(t) = RI_{in}(t) = R \sum_j^p I_b(t) \quad (\text{Eq. 3.9})$$

where $\tau_m = RC$ of leaky integrator [5] and p denotes the number of branches connected to the neuron. Whenever $V_m > V_{th}$, the neuron asynchronously fires a post-synaptic spike S_{post} of duration t_p and then resets to zero.

$$\text{If } V_m(t) \geq V_{th} \begin{cases} S_{post}^{(\tau)} \rightarrow 1 & t \leq \tau < t + t_p \\ S_{post}^{(\tau)} \rightarrow 0 & \tau = t + t_p \end{cases} \quad (\text{Eq. 3.10})$$

The I&F neuron circuit from [50] is used for SPICE simulation, with the memory trace $T_{pre/post}$ generated by

$$T_{pre/post} = K(t) * S_{pre/post} \quad (\text{Eq. 3.11})$$

where $K(t)$ is the spike convolution kernel. The circuit is thus Globally Asynchronous Locally Synchronous since the dendritic branch outputs are synchronous

but the neuronal firings are asynchronous.

Connections between silicon neurons located on different chips are essential for building even moderately sized artificial neural systems. For enabling connections between neurons in different chips address event representation (AER) is widely used. AER is used to transmit neuronal spikes, which are clustered in space (neuron address) and time, on a time-multiplexed communication channel [51]. The I&F circuit used (see Fig. 3.12(b)), generates a REQ signal along with the spike. The REQ signal from the sender intimates the receiver of data (address) transfer. Upon successfully receiving the data, receiver pulls up an active low acknowledgment signal (ACK). This ACK signal resets the V_m to ground. Fig. 3.12(a) shows evolution of V_m with detection of MTJ output. The receiver can be in other IC or test setup. AER serves as an effective way to arbitrate fewer communication buses for large neuromorphic circuits.

3.4 Results

3.4.1 Tunable STDP curves with Change in Circuit Parameters

The STDP curves obtained from the DW synapse are shown in Fig. 3.13(a). The STDP curves obtained from the DW Magnet based synapse are tunable according to T_{pre} and T_{post} and maximum write current (I_{max}). It is observed that an increase in I_{max} increases ΔW_{max} as shown in Fig. 3.13(b). Fig. 3.13(c) shows STDP curves for different values of ϵ (ratio of T_{pre} and T_{post}). By interchanging T_{pre} with T_{post} and S_{pre} with S_{post} in Fig. 3.10(c), we can get Reverse STDP (R-STDP) curve shown in Fig. 3.13(d). In R-STDP, pre-synaptic spike arrival after post-synaptic spike arrival increase corresponding synaptic weights. And post-synaptic spike arrival after pre-synaptic spike arrival depresses the corresponding weights.

With an I_{max} of $45\mu\text{A}$ the average write energy/spike requirement is estimated to be only 96fJ from SPICE simulations on a 65nm CMOS process. Table 3.2 shows

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

Figure 3.13: **tunable STDP curves** a) STDP curve obtained from SPICE simulation of the circuit shown in Fig. 3.10. b) ΔW_{max} increases with I_{max} , where I_{max} is the write current corresponding to $\Delta t = 0$. c) STDP curve obtained for ratio of time windows $(T_{\text{pre}}/T_{\text{post}}) \in < 1$. d) R-STDP curve obtained by interchanging T_{pre} and S_{post} with T_{post} and S_{pre} respectively in Fig. 3.10(c)

improvement in write energy compared to other plastic synapse implementing STDP learning rule. Note that the calculations in Table 3.2 are normalized for a $\Delta W_{\text{max}}/W$ of 0.25. Also the proposed neuromorphic circuit can work with lower power supplies compared to other synapse implementations. The calculation of energy required per spike for different implementations is explained in Appendix B.

Table 3.2: Comparison of Write Energy/spike

Reference	Type of synapse	V_{write}	Write - Energy/spike	Dynamic Range $\Delta W/W$	CMOS Process	Benchmark Task
<i>M.Prezioso et al</i> (2015) [52]	Memristor based synapse	$\pm 1.3\text{V}$	36nJ	100	-	Digital Image Classification (3×3)
<i>Y.Wu et al</i> (2012) [53]	Memristor based synapse	$\pm 1.5\text{V}$	4.2uJ	100	-	-
<i>S.H.Jo et al</i> (2010) [39]	Memristor based synapse	+3.2V / -2.9V	30pJ	-	-	-
<i>S.Brink et al</i> (2013) [19]	Floating gate based synapse	12V (Tunnel) 4.5V (Injection)	4.5pJ	-	350nm	Winner Take All (WTA) network
This work	DW-magnet based synapse	0.8V	96fJ	16	65nm	Hand Written Digit classification

3.4.2 Network Simulation: Handwritten Digit Classification

In this section, we look into classifying handwritten digits from the MNIST database [26] to demonstrate the online learning capability of our system. Let us first throw some light on the structure of the employed classifier. Typically, if we have a dataset having C classes of d dimensional input patterns, then the proposed classifier, depicted in Fig. 3.14, has C excitatory neurons with each neuron having m perceptrons - each equipped with $k \geq d/m$ synapses. The d input dimensions are randomly distributed across the synapses of an excitatory neuron. All the excitatory neurons have the

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

same network topology. The output of the j^{th} perceptron connected to the n^{th} neuron

Figure 3.14: **Network architecture used for simulation.** T is the time duration of one pattern.

Figure 3.15: **Weight update rule for semi-supervised learning:**The weight update for the synapses of neuron N_n when it is (a) supposed to respond to the current pattern but it remains silent, (b) supposed to remain silent but responds and (c) supposed to respond and it does so. The first two figures show how our learning rule handles error and the third figure shows how homeostasis or stop-learning is achieved.

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

is given by:

$$per_j^n = g_j\left(\sum_{i=1}^k W_i T_{pre}^i(t)\right) \quad (\text{Eq. 3.12})$$

where g_j is the sign function and same for all perceptrons and W_i and $T_{pre}^i(t)$ are the weights and inputs corresponding to the k input dimensions connected to its k synapses. The total input current received by the n^{th} neuron is thus given by:

$$I_{in}^n(t) = I_{scale} * \sum_{j=1}^m per_j^n \quad (\text{Eq. 3.13})$$

where I_{scale} is the scaling factor. I_{scale} can be tuned by sizing the transistor M_1 shown in Fig. 3.11(b). For the task at hand, we had $d = 784$, hence we have set $m = k = 28$. The network also has an inhibitory neuron which provides a global inhibitory current to all the excitatory neurons whenever any one of them emits an output spike. We employ a R-STDP based learning rule to train this network, which involves injecting a supervised current I_n^s to the n^{th} neuron such that it responds (emits a spike) when the network is excited by an input from the n^{th} class. The input images are converted to binary vectors/patterns which are subsequently transformed into spike trains by changing the 1's and the 0's by single spikes at t_{pre}^p and t_{pre}^q respectively given by:

$$\begin{aligned} \text{1's of the binary image: } S_{pre}^p &= \delta(t - t_{pre}^p) \\ \text{0's of the binary image: } S_{pre}^q &= \delta(t - t_{pre}^q) \end{aligned} \quad (\text{Eq. 3.14})$$

Note that we keep $t_{pre}^p < t_{pre}^q$ as this favors our learning process. In R-STDP, synaptic weight depression happens when a post-synaptic spike follows a pre-synaptic spike and potentiation occurs when a pre-synaptic spike comes after a post-synaptic spike. Evidences of such R-STDP rules have been widely reported in biological measurements [54]. Let us first look into the scenario when an error occurs in detecting the current pattern. Say neuron N_n is supposed to respond to the current pattern. The supervisory current I_n^s injected to N_n consists of a strong pulse at the beginning of the pattern which forces N_n to produce a spike during the duration of pulse.

All the remaining neurons receive a zero supervisory current. We consider a neuron has responded to a particular pattern if it produces a spike after the end of

Figure 3.16: **Receptive fields** :Receptive fields obtained for hand written digits 0 and 1, after training with network architecture shown in Fig. 3.14.

the pulse. The post-synaptic spike train of neuron N_n is given by S_{post} . If neuron N_n does not respond then weights of the synapses connected to the S_{pre}^p increases as the incoming pre-synaptic spike comes at a later time with respect to the post-synaptic spike. This mechanism, depicted in Fig. 3.15(a), trains N_n to detect the current class of pattern. The amount of weight change is given by:

$$\Delta W_n^p = T_{post}(t_{pre}^p) \quad (\text{Eq. 3.15})$$

Error may also occur when neuron N_n is supposed to be silent for the current pattern but it still responds. In this case as the post-synaptic spike follows the pre-synaptic spike, weights of the synapses connected to the S_{pre}^p decreases as shown in Fig. 3.15(b). This mechanism trains N_n to remain silent for subsequent patterns of the current class. The quantity of weight change is given by:

$$\Delta W_n^p = -T_{pre}^p(t_{post}) \quad (\text{Eq. 3.16})$$

On the other hand, two cases may arise when the network does a correct classification. In the first case, neuron N_n is supposed to remain silent and it does not respond and hence no weight updates take place. In the second case, neuron N_n is supposed to detect the current pattern and it responds thereby inducing a positive and a negative weight change for the synapses connected to S_{pre}^p to maintain homeostasis - controlled by the parameter γ . The net weight change is given by:

$$\Delta W_n^p = T_{post}(t_{pre}^p) - \gamma T_{pre}^p(t_{post2}) \quad (\text{Eq. 3.17})$$

where the parameter γ is used to set the net weight update to zero.

CHAPTER 3. POST-CMOS SOLUTION FOR SELF-LEARNING LOW-POWER SYNAPSE

After training the network for 3 epochs, the resultant receptive field formed is shown in Fig. 3.16. Moreover, the testing accuracy averaged over 10 trials for classifying different handwritten digit-pairs into two classes is 95.4667%(SD=2.948 %).

Chapter 4

Algorithm for Improvement of Spin-mode Neuromorphic Circuit

Previous chapter discussed a Spin-mode neuromorphic circuit for low-power implementation of neural network. We also saw the online-learning capability of network and its application in recognition of handwritten digit pairs. But the basic circuit shown in Fig. 3.6, suffers a practical limitation by increasing the number of synapses per MTJ neuron (k). This is due to the limited spin diffusion length of the non-magnetic copper channel beneath the DWM synapse. The spin current generated from input DW magnets decays exponentially as $\exp(-\frac{L}{\lambda_n})$ along the length of propagation 'L'. As the distance between input magnet and MTJ neuron increases the strength of input charge current I' should be increased to maintain constant effective net spin current I_{net}^{spin} (see section 3.3.3). The spin diffusion length of a copper channel is less than $1\mu\text{m}$. As we increase k , the strength of spin polarized current from DWM synapses needs to be increased. Hence for including higher number of synapse per branch, larger input charge current is required for each synapse. This can be seen clearly understood from the of circuit layout in Fig. 4.1a). Fig. 4.1b) shows the amount of excess current required for higher value of k .

CHAPTER 4. ALGORITHM FOR IMPROVEMENT OF SPIN-MODE NEUROMORPHIC CIRCUIT

Figure 4.1: **Layout of neuromorphic circuit** : Increasing number of synapses per branch of neural network would reduce the spin current available beneath MTJ free-layer. It can be noticed from the top-view of the synapse-neuron module, that decreasing length of DW-Magnets will be also useful to avoid the bottleneck of shorter spin-flip length.

Here we perform **a design space exploration of networks capable of producing significant results with few synapses per branch**. Further, we also try to find the optimum number of bits to which each synapse should be represented for maximizing the network performance. Decreasing number of bits in synapse can help us reduce the size of synapse (see section 3.3.2). The number of notches in a DW strip increases exponentially with number of bits in the synapse. Compact synapse size is favorable for vast integration and producing higher coherent spin currents beneath MTJ free-layer. In this context dendrites and branch are used interchangeably. [55] shows computational capability and capacity of network with active (non-linear) parallel dendrites. *Mel. et.al.* combined structural plasticity along with integration of non-linear dendritic compartments. Here summation of weighed inputs are subjected to non-linear activation function. The inputs of dendrites are replaced according to a leaning algorithm defined later.

4.1 Structural plasticity Algorithm

Structure of Network : The network specified in [55] compose of cell structures explained below. The forward propagation for a classification task is explained here. Each cell is made up of 'p' branches. The responses of a branch B_j can be written as follows.

$$b_j = B\left(\sum_{i=1}^K W_{ij}X_{ij}\right) \quad (\text{Eq. 4.1})$$

where W_{ij} is the weight of synapse in i-th branch and X_{ij} is input of synapse in i-th branch of a cell. These b_j responses are combined by summation in a cell to produce $a(x)$ the cell's response. The network is composed of C cells, for a C class classification. The class corresponding to maximum cell response is classified as output class . In light of this restriction to positive synaptic weight that result in values, cells can be split to give $a^+(\cdot)$ and $a^-(\cdot)$ response. $y(x) = \text{sgn}[a^+(x) - a^-(x)]$, where the sign operator maps the combined activity into one of two response categories (+, -). The i-th negative cell $a_i^-(x)$ in training is forced to give response when the training sample does not belong to class c_i .

Structural plasticity Algorithm : Each cell is initiated with random weights and random connectivity matrix. Connectivity matrix defines the index of input in each cell. Two types of modifications are possible in the structure. The weight of an existing synapse can be modified (weight change). Second, the structure defined by a connection of one input field to a specific dendrite in the cell can be modified with (structural change).The following methodology engages a batch mode learning for structural plasticity. After each pass through the training set, a random set 'T' of n_T synapses was considered for replacement. For each synapse in T, a fitness parameter [55] is computed using a standard delta rule which measured " the degree to which the synapse contributed to the overall performance " of the classification capability of the cell. The fitness Φ_{ij} for the i-th synapse in the j_{th} branch was given by a product of

four terms:

$$\Phi_{ij} = \langle X_{ij} b'_j(x) g'(y) (t - g(y)) \rangle \quad (\text{Eq. 4.2})$$

where the brackets $\langle \rangle$, indicate the average value over the training set, x_{ij} is the pre-synaptic activity, b'_j is the derivative of the post-synaptic branch response. The $g()$ can be a linear function or a binary function with $g'(y)$ being its derivative. Post-synaptic branch response $b()$ is a quadratic non-linearity. The synapse with minimum value of Φ in 'T' was tagged for reduction in synaptic strength by one. If the weight of the synapse become zero by this reduction, then it is replaced with best-performing synapse in a randomly chosen replacement set 'R' with cardinality n_R from 'd' input lines. Also the weight of the best performing synapse from the target set is increasing by one after each epoch. All weights are limited between 1 and l , where $l=2^b$. 'b' is number of bits. The replacement set can be considered as a set of dormant input fields unable to contribute to the cell activity. Upon finding the dormant input field strongly correlated with a specific branch activity, it is inducted into the branch for better structural stability. A momentum term (η) can be added to fitness parameter calculated in learning. It facilitates better optimization. It is incorporated as follows in fitness parameter:

$$\Phi_i = \Phi_i + \eta \Phi_{i-1} \quad (\text{Eq. 4.3})$$

In iterative learning methods inclusion of momentum can smooth out variations and will help in faster convergence.

4.2 Experiment and Result

Classification of hand-written digits were conducted with MNIST [26]. Three classes of MNIST database are considered for analyzing test performance with varying number of branches per cell. Each 28 X 28 images were compressed to 14 X 14 images by sub-sampling grids explained in section 2.1.1. Input fields of images are truncated to one bit by suitable threshold. The network produced following result (see Fig.4.2) for classification. Note that the results shown are for a network with single cell per

CHAPTER 4. ALGORITHM FOR IMPROVEMENT OF SPIN-MODE NEUROMORPHIC CIRCUIT

Figure 4.2: **Testing accuracy of network:** The result shows increase in network performance with increase in number of perceptrons per cell (m). Also the results show improvement in performance with higher number of bits in synapse. the difference in testing accuracy diminishes for larger m .

class. The results show improvement in network accuracy by using multi-bit synapses. Significant MNIST classification results can be produced by increasing number of cells per class. Moreover the network employs only 8 synapses per branch ($k=8$) for achieving results shown in Fig. 4.2. This reduces the requirement of sourcing excess charge current I' for each input synapse. We can understand from the result that even with 3 bit long synapses, we can classify patterns by increasing number of non-linear dendritic branches per class. This way we can design a neuromorphic circuit within geometry required to overcome the limitation produced by spin diffusion length.

Chapter 5

Conclusion and Future work

The thesis presents novel methods and hardware architectures for acceleration of neural networks to achieve higher energy efficiency. A multilayer neural network has been realized with compact single transistor synapses. We implemented a simple, low-power technique for feature extraction from images. This work maps RP²CNN, a convolutional neural network architecture, to single stage random projection core presented by [33]. With this proposed method, we can achieve up to 2 Tera-Ops/joule (see section 2.2.1), which is 50X energy efficient compared to recent hardware implementation of CNN like [3].

With co-simulation of circuit elements and spin transport modules we are able to apply STDP to Domain Wall Magnet synapses [9]. The proposed neuromorphic circuit could operate with power supply as low as 800mV. The write energy required per spike is 100 times less than other implementations in past (see table 3.2). The network is able to achieve significant performance in classification of handwritten digit pairs (see section 3.4.2), in spite of having only 4 bit precise synapses.

Chapter 4 discussed a possible neural-network configuration for spin mode neuromorphic circuit presented earlier. The design space exploration done here is useful for alleviating the limitation of spin diffusion length in spin-mode neuromorphic

CHAPTER 5. CONCLUSION AND FUTURE WORK

circuit (in Fig. 3.11). It is found that the bottleneck created by short spin diffusion length can be alleviated with non-linear dendritic structures with limited number of synapses per dendritic branch.

Future Work : *Chapter 2:* The random projection technique RP^2CNN explained in chapter 2 promises competitive result compared to [25] in Matlab simulations. Future research in the topic should find solutions to following short-comings. The effective weight of the synapse is defined by $e^{\frac{\Delta V_{T1}}{U_T}} - e^{\frac{\Delta V_{T2}}{U_T}}$. Since U_T , the thermal voltage and ΔV_{Ti} , the threshold variables are dependent on temperature, weight is also temperature dependent. We need to stabilize the random projection core response with respect to temperature fluctuations. The hardware implementation of the feature extraction module is also degraded by non-linearity of neuron responses as defined. As discussed in 2.2.1 the counter output is given by Eq. 2.9. The CCO-ADC responses according to RP^2CNN method is given by

$$\begin{aligned} f'_{cycle} &= \frac{I_{in}^i(I_{rst} - I_{in}^i)}{I_{rst} \cdot C_{neu} \cdot VDD} - \frac{I_{in}^{i+1}(I_{rst} - I_{in}^{i+1})}{I_{rst} \cdot C_{neu} \cdot VDD} \\ &= \frac{(I_{in}^i - I_{in}^{i+1})(I_{rst} - (I_{in}^i + I_{in}^{i+1}))}{I_{rst} \cdot C_{neu} \cdot VDD} \end{aligned} \quad (\text{Eq. 5.1})$$

Here linear approximation is difficult to hold, since the sum of current drawn by two adjacent columns $(I_{in}^i + I_{in}^{i+1})$ will be a comparable fraction of I_{rst} . As explained by Jarret *et al* [25], the algorithm can be further improved by adding a gain parameter to non-linearity function in each neuron. By increasing the linearity of CCO response we can accommodate this adaptive gain parameter in the hidden neuron for better image recognition.

Chapter 3: Below mentioned are the scope of future research in spin-based neuromorphic circuit. The simulation framework used for DW motion along ferromagnetic nanowire does not consider nonidealities like surface roughness, stochastic thermal effects. The DW motion can experience issues such as change in chirality, thermally activated depinning of DW from notches [56]. As explained in [57] different vortex con-

CHAPTER 5. CONCLUSION AND FUTURE WORK

figuration of domain wall are depinned at different current densities. Hence, the future research in implementation of synaptic plasticity with DW magnets should consider these effects.

Chapter 4: The network presented in chapter 4 employs a quadratic branch non-linearity 'B' (see equation Eq. 4.1). This should be simplified to a Heaviside function within the capability of spin-mode neuromorphic circuit. The future work in this topic also includes classification of entire MNIST database with the help of improved structural plasticity algorithms.

Appendix

.A Simulation Setup Used for Spin-Mode Neuromorphic Circuit

Figure .1: **Simulation framework** : Simulation framework employed in this work is similar to the one in [9]. Block (1), which mimics the DW motion under the influence of current pulse is explained by equation (Eq. 3.5) and (Eq. 3.6). The same is simulated using Verilog A model.

Fig. .1 shows the simulation framework employed in this work. The 1-D model explains DW motion dynamics and hence position of Domain wall after release of current pulse. The domain wall magnet strip could be broken into two different domains of oppositely magnetized ferro-magnets, as dictated by (W) the position of Domain wall magnet. Hence the structure turns out to be a multi-input LSV structure (see Fig .2) similar to [11]. Spin-magnet circuits [11] approach leads to the mapping of a spin device structure, involving nano-magnets interacting through nonlocal spin transport, into an equivalent spin circuit. By following spin-magnet circuits approach, the spin-transport models yields four-component conductance models. G_{FM-NM} , G_{FM} and

CHAPTER 5. CONCLUSION AND FUTURE WORK

G_{NM} for the elements, *magnetchannel interface*, *ferro-magnet* and the *nonmagnetic channel*, respectively. These spin-transport models form block (3) of Fig. .1.

A thin MgO interface is added in the structure, between channel and DW magnet to increase spin polarization of injected electrons [11]. Each of these modules look like a Pi-network as shown in Fig. .3. The series component represents the usual conductance of the material and shunt terms are included to account for loss of spin currents as they travel through a material, explained by spin-flip . These conductance are four component conductance matrices with charge component c and the three spin components x, y and z . The conductance matrices relate four-component voltage drop and current flow between different circuit nodes by equation 1.

$$\begin{bmatrix} I_c \\ I_x \\ I_y \\ I_z \end{bmatrix} = G_{c,x,y,z} [4 \times 4] \begin{bmatrix} V_c & V_x & V_y & V_z \end{bmatrix} \quad (\text{Eq. .2})$$

After solving for spin currents in the channel, these spin currents are used to evaluate output nano-magnets magnetization. Block(2) of Fig. .1, which gives the instanta-

Figure .2: **Multi-input LSV structure** [11]: Magnetization dynamics and spin-transport model of all ferromagnets and magnetchannel interface will be self-consistently simulated. The magnetization of the input ferromagnets (partitioned by DW) will be obtained from 1-D model as explained in section 3.3.3. The model shown is interpolated to 2-D plane as explained in [9].

CHAPTER 5. CONCLUSION AND FUTURE WORK

neous magnetization of a mono-domain ferro-magnet, under the action of spin current is given by the following Landau-Lifshitz-Gilbert (LLG) equation.

$$\frac{(1 + \alpha^2)}{\gamma_0 H_k} \frac{d\hat{m}}{dt} = \hat{m} \times h - \alpha \hat{m} \times (\hat{m} \times h) - \hat{m} \times (\hat{m} \times i_s) - \alpha \hat{m} \times i_s \quad (\text{Eq. .3})$$

where, h is the sum of all magnetic fields, i_s is the spin current (scaled by $(1)/(2qN_s)$), and $\hat{m}$ is the instantaneous magnetization of the ferromagnet, γ_0 is the gyro-magnetic ratio and N_s is the number of spins in a nanomagnet.

The conductance matrices of magnetchannel interface and ferromagnet are dependent upon the magnetization state of the grid points, and hence, the spin-diffusion transport is solved self-consistently with LLG equation. The spin circuit is benchmarked with [58]. The simulation structure is extended to 2 dimensions as explained in [9], where the area of input ferromagnets depends upon W . The Verilog A models for above mentioned conductance models and LLG where obtained from [8], [44]. Results are obtained in this work by combined simulation of these models, and write circuitry (see Fig. 3.10) in SPICE platform. An abstract model was extracted from the SPICE simulation results . A complete Neural Network, with these abstract models of synapses and neurons were simulated in Matlab.

Figure .3: π **Conductance model** : Pi conductance model for G_{FM-NM} , G_{FM} and G_{NM}

.B Calculation of Write Energy per Spike

The resistance of memristors are programmed by voltage pulses. The memristor changes its conductance from $G_{initial}$ to G_{final} through the course of voltage pulses. In the write energy /spike calculation of Y.Wu *et.al.* [53] a write voltage (V_W) of $\pm 1.5V$ is used. For a controlled change of conductance of memristor, $500\mu s$ wide SET and RESET pulses are used. 12 SET/RESET pulses are required for changing the associated weight (ΔW) by 25%. Considering an average conductance G_{avg} of $3.1 \times 10^{-4} S$ the power consumption is 0.7mW. By multiplying power with time duration yields $4.2\mu J$

Mpreizo et.al. implemented a digital image classification with 60 memristors. The memristors were programmed with (V_W) of $\pm 1.3V$. 75 pulses of width $10\mu s$ (minimum width) were used to change conductance of memristor by 25%. The average conductance of memristors between highest and lowest conductance was $35\mu S$. By following same calculations as above, the write energy /spike for [52] is 36nJ.

S.H.Jo *et.al.* [39] implements a programmable synapse with V_W of $3.2V/-2.9V$. An average current of 25nA is drawn for $400\mu s$. Hence this implementation of synapse consumes 30pJ for an average modification of weight by 25% from any initial weight (W_0).

The write energy per spike in Floating Gate synapse implementation [19] is dominated by injection. Injection phase need a potential of 4.5V. An injection current of 1nA for 1ms consumes 4.5pJ. In this work the write energy per spike is calculated as below. Modification of weight at each spike instance involves one part of transistor stack in Fig.3.10(c). The average magnitude of write current I_w over T_{pre} memory trace is $15\mu A$. For an average change ΔW by 25% i.e 4 units (see Fig.3.7(b)), 63 fJ is required. Note that I_w flows across ΔV for only 100ns (t_p). Apart from this the MOS gate capacitance of transistor pair ($M_3 - M_4$ or $M_5 - M_6$) consumes 23fJ.

Publication

1. G. Narasimman, S. Roy, X. Fong, K. Roy, C. H. Chang and A. Basu, "A low-voltage, low power STDP synapse implementation using domain-wall magnets for spiking neural networks," 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, 2016, pp. 914-917.

Bibliography

- [1] IBSG Cisco and D Evans. The internet of things how the next evolution of the internet is changing everything, 2011.
- [2] M A Ranzato, F J Huang, Y-L Boureau, and Y LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [3] V. Gokhale, J. Jin, A. Dunder, B. Martini, and E. Culurciello. A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 682–687, 2014.
- [4] Y Chen, E Yao, and A Basu. A 128 Channel 290 Gmacs/w Machine Learning Based Co-processor For Intention Decoding In Brain Machine Interfaces. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pages 3004–3007. IEEE, 2015.
- [5] J. P. Pfister. *Theory of Non-Linear Spike-Time Dependent Plasticity*. PhD thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland, 2006.
- [6] E. Chicca G. Indiveri and R. Douglas. A VLSI Array of Low-power Spiking Neurons and Bistable Synapses with Spike-timing Dependent Plasticity. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 17:211–221, 2006.
- [7] S. Ramakrishnan, P E. Hasler, and C. Gordon. Floating Gate Synapses with Spike-Time-Dependent Plasticity. *Biomedical Circuits and Systems, IEEE Trans-*

BIBLIOGRAPHY

- actions on*, 5(3):244–252, 2011.
- [8] S Datta, S Salahuddin, and B Behin-Aein. Non-volatile Spin Switch For Boolean And Non-boolean Logic. *Applied Physics Letters*, 101(25):252411, 2012.
- [9] M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy. Spin-based Neuron Model With Domain-wall Magnets As Synapse. *Nanotechnology, IEEE Transactions on*, 11(4):843–853, 2012.
- [10] http://www.scholarpedia.org/article/Silicon_neurons.
- [11] S. Srinivasan, V. Diep, B. Behin-Aein, A. Sarkar, and S. Datta. Modeling multi-magnet networks interacting via spin currents. *arXiv preprint arXiv:1304.0742*, 2013.
- [12] A Muthuramalingam, S Himavathi, and E Srinivasan. Neural Network Implementation Using FPGA: Issues and Application. *International journal of information technology*, 4(2):86–92, 2008.
- [13] W Gerstner and W M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.
- [14] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.
- [15] G. Indiveri, E. Chicca, and R. Douglas. A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses With Spike-Timing Dependent Plasticity. *IEEE Transactions on Neural Networks*, 17(1):211–221, Jan. 2006.
- [16] M. Mahowald. *VLSI analogs of neuronal visual processing: a synthesis of form and function*. PhD thesis, California Institute of Technology, 1992.
- [17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [18] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camuñas-Mesa, R. Berner, M. Rivas-Pérez, T. Delbrück, et al. CAVIAR: A 45k neuron, 5M synapse, 12G connects/s

BIBLIOGRAPHY

- AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking. *Neural Networks, IEEE Transactions on*, 20(9):1417–1438, 2009.
- [19] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan. A learning-enabled neuron array IC based upon transistor channel models of biological phenomenon. *IEEE Transactions on Biomedical Circuits and Systems*, 7(1):71–81, Feb. 2013.
- [20] C Zamarreño-Ramos, L A Camuñas-Mesa, J A Pérez-Carrasco, T Masquelier, T Serrano-Gotarredona, and B Linares-Barranco. On Spike-timing-dependent-plasticity, Memristive Devices, And Building A Self-learning Visual Cortex. *Frontiers in neuroscience*, 5, 2011.
- [21] M. S. Fashami, K. Roy, J. Atulasimha, and S. Bandyopadhyay. Magnetization Dynamics, Bennett Clocking and Associated Energy Dissipation in Multiferroic Logic. *Nanotechnology*, 22(15):155201, 2011.
- [22] S S-P Parkin, C Kaiser, A Panchula, P M Rice, B Hughes, M Samant, and S-H Yang. Giant Tunnelling Magnetoresistance at Room temperature with MgO (100) Tunnel Barriers. *Nature materials*, 3(12):862–867, 2004.
- [23] E Martinez, L Lopez-Diaz, O Alejos, and L Torres. Thermally Activated Domain Wall Depinning In Thin Strips With High Perpendicular Magnetocrystalline Anisotropy. *Journal of Applied Physics*, 106(4):043914, 2009.
- [24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based Learning Applied To Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [25] K Jarrett, K Kavukcuoglu, M A Ranzato, and Y LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.
- [26] <http://yann.lecun.com/exdb/mnist/>.
- [27] G. Griffin, A. Holub, and P. Perona. Caltech-256 Object Category Dataset. 2007.
- [28] A Saxe, P W Koh, Z Chen, M Bhand, B Suresh, and A Y Ng. On Random Weights

BIBLIOGRAPHY

- and Unsupervised Feature Learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1089–1096, 2011.
- [29] G.-B. Huang, Q.-Y Zhu, and C-K Siew. Extreme Learning Machine: Theory And Applications. *Neurocomputing*, 70(1):489–501, 2006.
- [30] E. Säckinger, B. E Boser, J. M Bromley, Y. LeCun, and L. D Jackel. Application of the anna neural network chip to high-speed character recognition. *IEEE Transactions on Neural Networks*, 3(3):498–505, 1992.
- [31] T. Behan, Z. Liao, L. Zhao, and C. Yang. Accelerating Integer Neural Networks on Low Cost DSPs. In *Proc. Int. Conf. on Intelligent Systems*, pages 1270–1273, 2008.
- [32] J. Fieres, A. Grübl, S. Philipp, K. Meier, J. Schemmel, and F. Schürmann. A Platform For Parallel Operation of VLSI Neural Networks. In *Proc. of the 2004 Brain Inspired Cognitive Systems Conference (BICS2004)*, 2004.
- [33] E. Yao and A. Basu. VLSI Extreme Learning Machine: A Design Space Exploration. *arXiv preprint arXiv:1605.00740*, 2016.
- [34] M. R. Azghadi, N. Iannella, S. F Al-Sarawi, G. Indiveri, and D. Abbott. Spike-based Synaptic Plasticity In Silicon: Design, Implementation, Application, And Challenges. *Proceedings of the IEEE*, 102(5):717–737, 2014.
- [35] W. Maass. On the computational Power of Noisy Spiking Neurons. *Advances in neural information processing systems*, pages 211–217, 1996.
- [36] Jilles Vreeken et al. Spiking Neural Networks, an Introduction. *Institute for Information and Computing Sciences, Utrecht University Technical Report UU-CS-2003-008*, 2002.
- [37] L F Abbott and S B Nelson. Synaptic plasticity: taming the beast. *Nature neuroscience*, 3:1178–1183, 2000.
- [38] Paul E. H. Shubha R. and Christal G.. Floating Gate Synapses with Spike-Time Dependent Plasticity. *IEEE Transactions on Biomedical Circuits and Systems*, 5, 2011.

BIBLIOGRAPHY

- [39] S. H. Jo, I. Chang, T. Ebong, B. B Bhadviya, P. Mazumder, and W. Lu. Nanoscale Memristor Device As Synapse In Neuromorphic Systems. *Nano letters*, 10(4):1297–1301, 2010.
- [40] NATURE NANOTECHNOLOGY. Memory With a Spin. 2015.
- [41] S. SP. Parkin, M. Hayashi, and L. Thomas. Magnetic Domain-wall Racetrack Memory. *Science*, 320(5873):190–194, 2008.
- [42] T. Yang, T. Kimura, and Y. Otani. Giant Spin-accumulation Signal And Pure Spin-current-induced Reversible Magnetization Switching. *Nature Physics*, 4(11):851–854, 2008.
- [43] E. Martinez, L. Lopez-Diaz, O. Alejos, L. Torres, and M. Carpentieri. Domain-wall Dynamics Driven By Short Pulses Along Thin Ferromagnetic Strips: Micromagnetic Simulations And Analytical Description. *Physical Review B*, 79(9):094430, 2009.
- [44] B. Behin-Aein, D. Datta, S. Salahuddin, and S. Datta. Proposal For An All-spin Logic Device With Built-in Memory. *Nature nanotechnology*, 5(4):266–270, 2010.
- [45] D. Lacour, J.A. Katine, L. Folks, T. Block, J.R. Childress, M.J. Carey, and B.A. Gurney. Experimental evidence of multiple stable locations for a domain wall trapped by a submicron notch. *Applied physics letters*, 84(11):1910–1912, 2004.
- [46] Eduardo Martinez. The Stochastic Nature Of The Domain Wall Motion Along High Perpendicular Anisotropy Strips With Surface Roughness. *Journal of Physics: Condensed Matter*, 24(2):024206, 2012.
- [47] HJ. van Driel. Models for Current-Driven Domain Wall Motion. *Utrecht University*, 2008.
- [48] G. Indiveri, E. Chicca, and R. Douglas. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *Neural Networks, IEEE Transactions on*, 17(1):211–221, 2006.
- [49] S Hussain, S.C Liu, and A Basu. Hardware-Amenable Structural Learning for Spike-Based Pattern Classification Using a Simple Model of Active Dendrites.

BIBLIOGRAPHY

- Neural computation*, 2015.
- [50] A. Banerjee, S. Kar, S. Roy, A. Bhaduri, and A. Basu. A Current-mode Spiking Neural Classifier with Lumped Dendritic Nonlinearity. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pages 714–717. IEEE, 2015.
 - [51] K. Boahen et al. Point-to-point Connectivity Between Neuromorphic Chips Using Address Events. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 47(5):416–434, 2000.
 - [52] M. Prezioso, F. Merrih-Bayat, BD Hoskins, GC Adam, K. K. Likharev, and D. B. Strukov. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521(7550):61–64, 2015.
 - [53] Y. Wu, S. Yu, H-S Philip Wong, Y.S. Chen, H.Y. Lee, S.M. Wang, P.Y. Gu, F. Chen, and M.J. Tsai. Alox-based resistive switching device with gradual resistance modulation for neuromorphic device application. In *Memory Workshop (IMW), 2012 4th IEEE International*, pages 1–4. IEEE, 2012.
 - [54] J J. Letzkus, B M. Kampa, and G J. Stuart. Learning Rules For Spike Timing-dependent Plasticity Depend On Dendritic Synapse Location. *The Journal of neuroscience*, 26(41):10420–10429, 2006.
 - [55] P. Poirazi and B. W. Mel. Impact Of Active Dendrites And Structural Plasticity On The Memory Capacity Of Neural Tissue. *Neuron*, 29(3):779–796, 2001.
 - [56] M. Laufenberg, D. Backes, D. Bührer, W. and Bedau, M. Kläui, U. Rüdiger, Carlos A F. Vaz, J A. C Bland, L J. Heyderman, F. Nolting, et al. Observation of Thermally Activated Domain Wall Transformations. *Applied physics letters*, 88(5):052507, 2006.
 - [57] DM Burn and D Atkinson. Effective Pinning Energy Landscape Perturbations for Propagating Magnetic Domain Walls. *Scientific Reports*, 6, 2016.
 - [58] B. Behin-Aein, A. Sarkar, S. Srinivasan, and S. Datta. Switching energy-delay of all spin logic devices. *Applied Physics Letters*, 98(12):123510, 2011.