An On-Chip NRW based Learning IC for Spiking Neural Network

Saurabh Dash[1], Sougata Kar[2], Arindam Basu[2] and Indrajit Chakrabarti[1]

***Abstract*—In the recent past there has been an increasing demand for area and energy efficient on-chip implementation of Machine Learning techniques. This paper presents a modification to the learning technique named Network Rewiring (NRW) for Dendritically Enhanced Readout (DER), the architecture and the various circuits required to implement on-chip learning. We show that with certain modifications, NRW can be implemented on-chip to classify patterns upto 10 classes with performance comparable to the software implementation. Furthermore, it is also shown that the design is tolerant to variation.**

I. Introduction

AI workloads on current hardware unavoidably spend a lot of time and energy moving data from memory to processing and back, stymieing their adoption in mobile devices. In this article, we will focus on an on-chip implementation of a recently proposed learning algorithm named Network Rewiring (NRW) learning rule for Dendritically Enhanced Readout (DER) [**??**] for spiking neural networks. This can be used for classifying spikes from a brain-machine interface or bio-inspired sensors[**??**] and uses binary synapses which help reduce the effect of statistical variations. Since these neural network chips could be deployed as embedded sensors in real world scenarios where the input data distribution is changing over time, on-chip learning bestows a degree a flexibility allowing the chip to accommodate this change in nature of data.

II. Background and Theory

In [**??**], DER has been used as the readout of Liquid State Machine known as Liquid State Machine with Dendritically Enhanced Readout (LSM-DER). In LSM-DER, the DER stage is responsible for extracting information from the liquid output and is trained using the NRW algorithm described in detail in [**??**] to perform binary classification and regression. Here, we present an overview of the same for the sake of completeness. The DER architecture is shown in Fig. 1. It employs two nonlinear neuronal cells (NL-cell), each of which has m identical dendritic branches connected to it with each branch having k excitatory synapses. If $\mathbf{x}$ is an input vector to this system, then each synapse is excited by any one of the d dimensions of the input vector where $d \gg k$. The output response of j^{th} dendritic branch is calculated as a nonlinear weighted sum of the k synaptic points connected to the branch and is given by $z_j = b(\sum_{i=1}^{k} w_{ij}x_{ij})$, where $b()$ is a model of the dendritic nonlinearity ($b(x) = x^2/x_{thr}$ for $x < x_{sat}$), w_{ij} is the synaptic weight of the i^{th} synapse on the j^{th} branch and x_{ij} is the input arriving at that particular synaptic connection. Combining all the dendritic responses, the overall output $f(x)$ of the neuronal cell is given by:

$$f(\mathbf{x}) = \sum_{j=1}^{m} z_j = \sum_{j=1}^{m} b(\sum_{i=1}^{k} w_{ij}x_{ij}) \tag{1}$$

where $f()$ denotes the neuronal current-frequency conversion function. The output of DER was calculated by noting the difference of the output of the two NL-cells. The overall output of the circuit is given by:

$$y = g[f_{+}(\mathbf{x}) - f_{-}(\mathbf{x})] \tag{2}$$

where the function g depends on the task. For example, it is a heaviside function for classification.

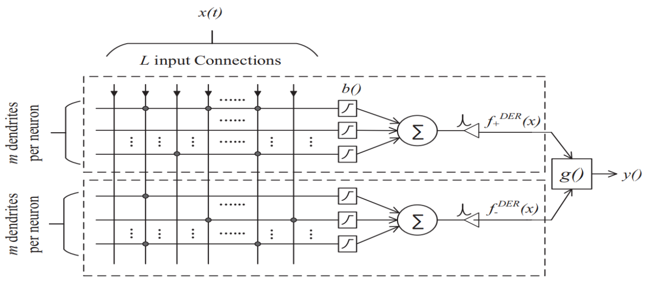


Fig. 1. Architecture of DER network.

III. Proposed Modifications

A. Branch-wise NRW with Adaptive Dropouts

The modified NRW algorithm to train DER primarily involves the following steps:

1) For connecting the input to the neuronal cells binary synapses have been used, thus weight changes imply connection changes. Since $d \gg k$, the learning chooses best k synapses for every branch.
2) At each step, the fitness index $c_{ij} =< x_{ij}b_j^{'}(t-y) >$ is calculated for all the synapses in the positive cell and $c_{ij} = - < x_{ij}b_j^{'}(t-y) >$ for the synapses in the negative cell. The synapse with the lowest c_{ij} in each branch is selected using a loser-take-all unit for

Saurabh Dash and Indrajit Chakrabarti are with the Department of Electronics and Electrical Communications Engineering, IIT Kharagpur (email: saurabhdash@iitkgp.ac.in, indrajit@ece.iitkgp.ac.in). Arindam Basu and Sougata Kar are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: arindam.basu@ntu.edu.sg) replacement unless the branch is dropped out from the replacement process with a certain probability p_{dr}.

3) The replacement synapse is chosen from a candidate set R, for each branch having n_R of the d input lines. The set R is created by placing n_R 'silent' synapses from d input lines.
4) Synaptic connections are modified if the replacement leads to a decrease in the error. If the error does not reduce even after 50 such iterations, the replacement is done to escape the 'local minima'.
5) The above steps are repeated until either all the test patterns have been memorized or 150 local minima have been encountered.

The probability of dropout can be kept constant but by varying the probability of dropout as a hyper-parameter, faster learning and lower accuracy can be obtained.

a) Determination of network hyper-parameter: Let, the probability of dropout be: p_{dr}
Probability of a branch chosen,

$$p = 1 - p_{dr} \tag{3}$$

Probability of at least 1 branch chosen,

$$p^{'} = 1 - (1 - p)^{m} \tag{4}$$

Suppose, we want $p^{'} \geq 0.999$. Implies,

$$p_{\infty} \geq 1 - (1 - p^{'})^{\frac{1}{m}} \tag{5}$$

We get, $p_{\infty} \geq 0.168$

$$p = p_{\infty} + \kappa(p - p_{\infty}) \tag{6}$$

The dropout rate (κ) was taken to be 0.8.

B. Results

The given results are obtained for 10,000 training and 8,000 testing samples from the standard MNIST dataset. In this article, the values of m,k,n_T,n_R were set to 25, 15, 15, 15 respectively for simulations, unless mentioned otherwise.

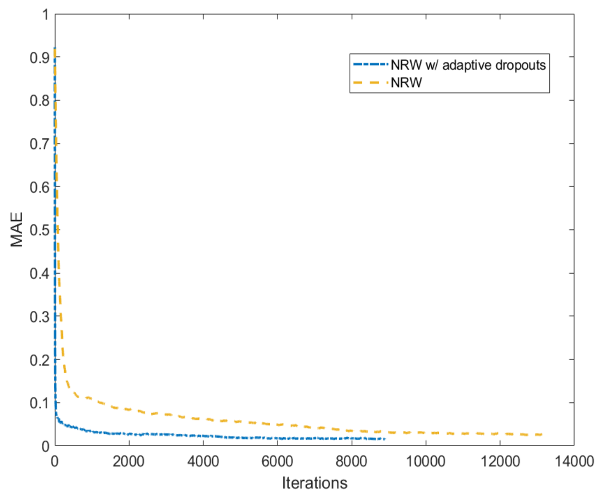


Fig. 2. Comparison of NRW and NRW w/ adaptive dropouts learning rule.

TABLE I
MEAN SQUARE ERROR

Learning Rule	Training Error	Testing Error
NRW	2.54%	6.40%
NRW w/ adaptive dropout	1.56%	5.80%

IV. MODIFICATIONS FOR VLSI IMPLEMENTATION

A very important practical constraint while implementing the above learning rule on hardware is the time constant of the averaging circuit. The order of time period is in milliseconds, which may be smaller than the the duration T_p of a single pattern encountered in the real world. Our proposed modification takes into account such cases as well. To reduce the number of hardware blocks in the chip, the input pattern size was decreased from 784 pixels to 196 pixels by average pooling. Instead of comparing all the synapses in a branch to identify the worst performing synapse in that branch, a subset of the synapses is chosen to reduce the number of correlation calculator and WTA/LTA blocks.

1) After presentation of P_{sub} number of patterns of the entire dataset P, a set L_{sub} is formed by choosing the input dimensions where at least, one spike has occurred. The initial connections of the cells are chosen from this set.
2) A target set T and a replacement set R is initialized from the set L_{sub}
3) A sub-target set T_{sub} and a sub-replacement set R_{sub}, subsets of T and R respectively is formed for each dendrite every t_{sp} seconds.
4) The averaging circuit finds $c_{ij}^{'} = sgn(t-y)\int_0^{t_{sp}} x_{ij}b_j^{'}$ for the synapses in T_{sub} and R_{sub}- an estimate of the true c_{ij} used in the earlier algorithm. The WTA and LTA circuits on each dendrite calculate the maximum c_{ij} from R_{sub} and minimum C_{ij} from T_{sub} respectively. This step has a loss in information since we no longer have information on the true c_{ij}.
5) To make up for the information lost in the earlier step, we introduce a voting mechanism to find the most probable maxima and minima. After presentation of P_{conn} number of patterns, the synapses chosen most number of times by the LTA circuits are chosen for replacement by the silent synapses on each branch which were voted most number of times by the WTA circuits, unless that branch is dropped out with a certain probability. Thus the network rewiring takes place after P_{conn} number of input patterns have been presented. The Mean Absolute Error (MAE) is checked for the next P_{MAE} patterns and if the average P_{MAE} does not decrease, the connections are switched back.
6) Steps 3-6 are repeated max_{iter} number of times after which the learning is terminated and the connection corresponding to best performance is saved as the final connection.

A. Results

The given results are obtained for 10,000 training and 8,000 testing samples from the standard MNIST dataset.The values of P_{conn}, P_{MAE} were set to P and P respectively for simulations, unless mentioned otherwise.

TABLE II
MEAN SQUARE ERROR

t_{sp}	Training Error	Testing Error
100	6.78%	8.81%
$P/4$	3.64%	6.39%

We can observe a 3% degradation in testing error due to the modifications made to the algorithm when the c_{ij} is calculated by averaging over 100 samples instead of 10,000.

V. IMPLEMENTATION OF THE ON-CHIP LEARNING IC

All the circuit simulations are done in a $180nm$ CMOS process.

A. Chip Architecture

Fig. 3 shows the architecture of the chip. This chip is suitable for up to ten class learning and classification. Each neuronal cell represents a class. While learning it will work in closed-loop, whereas classification is an open-loop operation. In each cell, there are m positive and m negative dendritic branches and each branch has k synapses. Additionally, there are n_R number of 'silent' synapses on every branch amongst which the best performing synapse is chosen. Instead of finding the worst performing synapse among all in a branch, we find the worst performing synapse from a subset of these synapses to reduce the number of c_{ij} blocks. This has been implemented using n_T number of 'copy' synapses per branch as shown in Fig. 3. These synapses do not contribute to the dendrite current. The inputs to the subset of interest are mirrored to these synapses which are connected to c_{ij} calculators and WTA/LTA blocks. The total number of some of the blocks required per cell is shown in Table III below.

TABLE III
NUMBER OF VARIOUS COMPONENTS PER CELL

Block	Number of Components
c_{ij} calculators	$2m \times n_T + 2m \times n_R$
Synapses	$2m \times k + 2m \times n_T + 2m \times n_R$
Squaring	$2m$

The input image is converted to a 1-D pulse train to be fed into the chip. 8-bit grayscale value are converted to 2-bit by thresholding. Pixels with value '1' contain spikes with a certain input frequency f while the pixels with value '0' contain no spikes. Although, the entire spike train is common to all the synapses, an input pixel is directed to the required synapse by using row and column decoders as shown in Fig. 3 which are driven by the connectivity matrix.

B. Building Blocks of the IC

a) Synapse: Synapse takes voltage pulses as its input and provides current pulses as output. Several synapse architectures exist which mimic the behaviour of biological synapse. Differential pair integrator (DPI) synapse [**??**] is one of the most popular synapse architectures which provides the control over threshold, time constant and weight of the synapse. The circuit schematic of the DPI synapse is shown in Fig. 5(a). In this circuit, transistors operate in sub-threshold region. The bias voltages V_w and V_τ are being set such that $I_w \gg I_\tau$ [**??**]. This converts a nonlinear circuit to a canonical first-order low pass filter. If an input spike of duration $t_i^+ - t_i^-$ is applied to the DPI synapse, the rise time of I_{SYN} is very small. The discharge profile can be modeled by the following equation.

$$I_{SYN}(t) = I_0 e^{-(\frac{t-t_i^+}{\tau_s})} \tag{7}$$

where $\tau_s = \frac{C_{SYN}U_T}{\kappa I_\tau}$, κ is the sub-threshold slope factor, and U_T is the thermal voltage. In [**??**] it has been derived and shown that steady state output current $\overline{I_{SYN}}$ (average of the output transient current) of the synapse is linearly related with the input pulse frequency f_{in} when it is excited with a continuous pulse train of average frequency f_{in} and pulse duration Δt.

$$\overline{I_{SYN}} = K'_{SYN} \times (\Delta t) \times f_{in} = K_{SYN} \times f_{in} \tag{8}$$

where K'_{SYN} is a constant set by V_w, V_{THR} and V_τ and $K_{SYN} = K'_{SYN} \times \Delta t$. Simulation results of the synapse is shown in Fig. 4 with the expected linearity. The maximum output current of the circuit is limited by the transistor M_{d6} driven by V_{SYN} as shown in 5(a) to be in sub-threshold. If $I_{THR,mos}$ is the threshold current of M_{d6}, then it can be written that

$$\overline{I_{SYN}} \leqslant I_{THR,mos} \implies f_{in} \leqslant \frac{I_{THR,mos}}{K_{SYN}} \tag{9}$$

b) Squaring: The current mode squaring circuit is shown in Fig. 5(b) as described in [**??**]. Transistors M_{S2}, M_{S1}, M_{S3} and M_{S5} form a translinear loop. The dendritic leak is implemented by adding transistor M_{S7} with a current I_{SLK} set by the bias V_{SLK}. This current I_{SLK} gets subtracted from the input current $I_{IN} = \overline{I_{SYN}}$ coming from the summed current of all the synapses in a dendritic branch. Hence the current through M_{S5} can be expressed as shown below in equation (10)

$$I_{SOUT} = \frac{(I_{IN} - I_{SLK})^2}{I_{STHR}} = \frac{(\overline{I_{SYN}} - I_{SLK})^2}{I_{STHR}} \tag{10}$$

I_{STHR} is the DC current through M_{S4} set by its gate voltage (V_{STHR}). Fig. 6 shows the simulation results of the square nonlinearity block when $I_{STHR} \approx 0$. Also, it can be seen that increasing V_{STHR} reduces I_{STHR} in equation (10) and increases the output current. The dynamic range of this circuit is limited by the requirement of keeping all the transistors in the translinear loop, in the sub-threshold region. Assuming $I_{SLK} \approx 0$ and the transistors in the translinear loop have

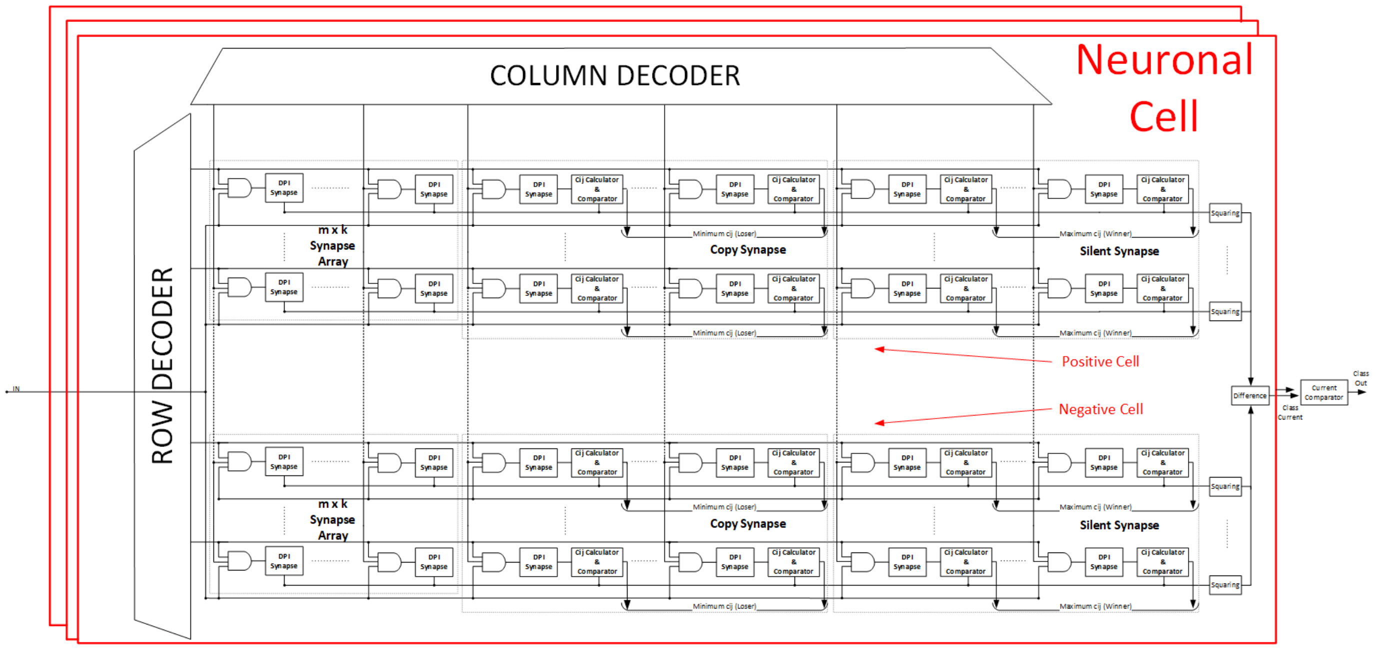


Fig. 3. Chip Architecture.

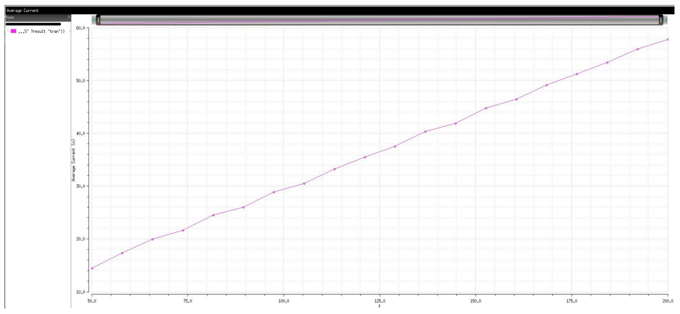


Fig. 4. Linear $\overline{I_{SYN}}$ vs f_{in} behaviour of DPI Synapse.

the same dimensions as the threshold current $I_{thr,mos}$, the constraint on average synaptic current entering this block can be written as:

$$\overline{I_{SYN}} \leqslant \sqrt{I_{STHR,max} I_{SOUT,max}} = I_{thr,mos} \tag{11}$$

It can be seen that the constraint set by equation (11) is the same as the one set by equation (9), if I_{STHR} can be set to the maximum value of $I_{thr,mos}$. The simulation result of this circuit and its comparison to the desired response can be seen in Fig. 6

c) Multiplier: The schematic of the current mode multiplier circuit is shown in Fig. 5(c). Its output current can be expressed as shown below

$$I_{out} = \frac{I_{IN1} \times I_{IN2}}{I_{THR}} \tag{12}$$

where $I_{IN1} = x_{ij}$, the individual synapse current, $I_{IN2} = b_j^{'}$, the respective branch current before squaring and I_{THR} is the normalizing current. Transistors M_{M1}, M_{M2}, M_{M3} and M_{M4} form a translinear loop. The simulation results are shown in Fig. 8. I_{IN2} was chosen to be $60nA$ and I_{thr} was chosen to be $30nA$. I_1 was swept from 10nA to 100nA. The ideal input-output plot is expected to be a straight line with a slope of 2. Operating range is given by the 10% deviation mark.

d) Current Mode LPF: The output of the multiplier circuit is averaged by a current mode LPF shown in Fig. 5(d). Time constant of this LPF is controlled by the current I_τ and the capacitor C. Lower I_τ gives a large time constant, increasing the averaging duration. Though there is a desire to have a large averaging time period, but this involves a direct trade-off with capacitor sizes as I_τ can not be lowered below a few pA. In this design, $I_\tau = 10pA$ and $C \approx 5pF$ leading to a time constant of $11.08ms$.

e) Current Mode LTA: The output current of the averaging LPF is passed through either WTA or LTA circuits for comparison. Schematic of the current mode LTA is shown in Fig. 5(e) along with the active synapse implementation. M_{L11} and M_{L12} along with I_{BIAS} constitute the basic LTA circuit [**??**]. Multiple LTAs are connected together with a common node V_C. The input currents are compared and the total I_{BIAS} current flows through the minimum input current LTA as its output. The output current of all other LTAs become zero. The output current of all the LTAs are then converted to either a '0' or a '1' through M_{L13}, M_{L14} and M_{L15}. The minimum input current LTA which takes away most of the I_{BIAS} is the loser and gives '1' whereas all other LTAs outputs remain '0'.

f) Current Mode WTA: The functionality of the WTA is just the opposite of the LTA, it detects the maximum current. Schematic of the current mode WTA circuit is shown in Fig. 5(f) in which multiple WTAs are connected together with a common node V_C. M_{W11} and M_{W12} along with I_{BIAS}

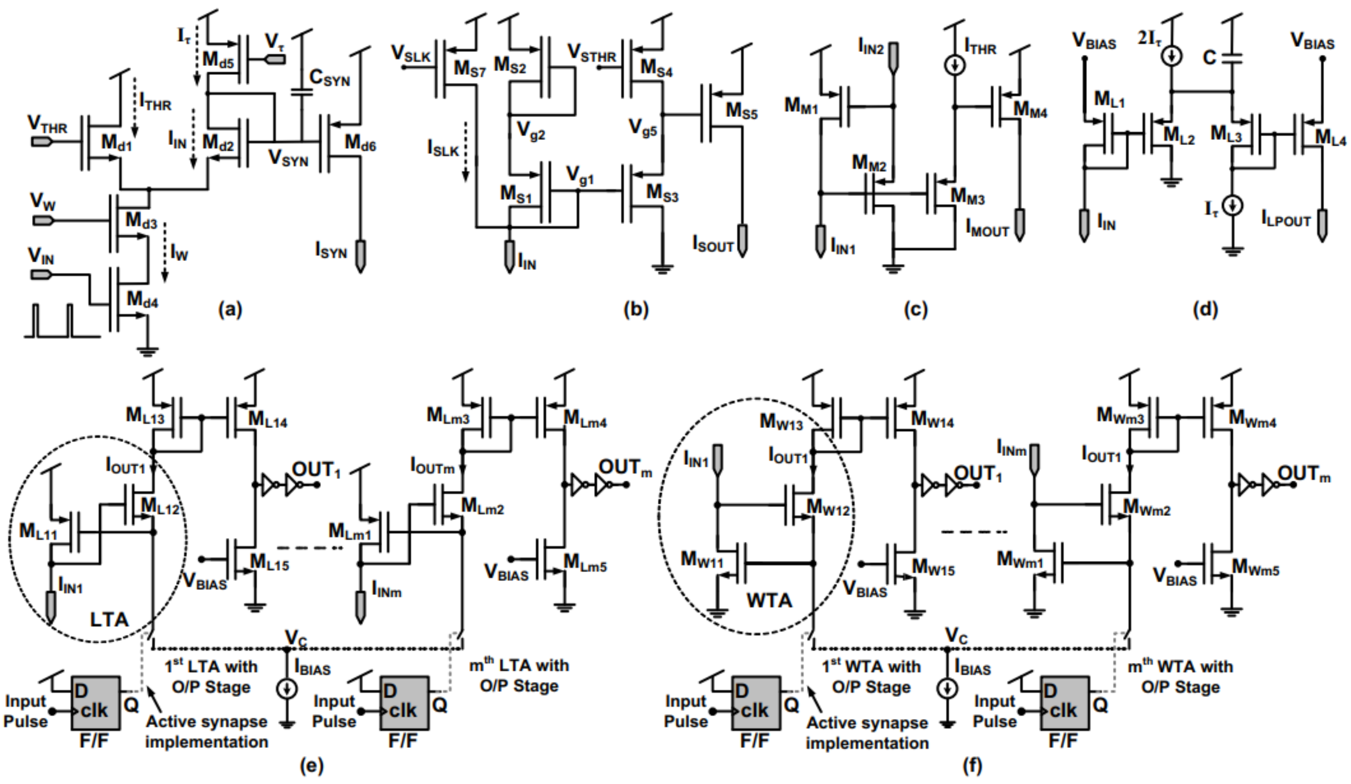


Fig. 5. (a) DPI Synapse, (b) Squaring Circuit, (c) Current Multiplier Circuit, (d) Current Mode LPF, (e) LTA Circuit, (f) WTA Circuit.

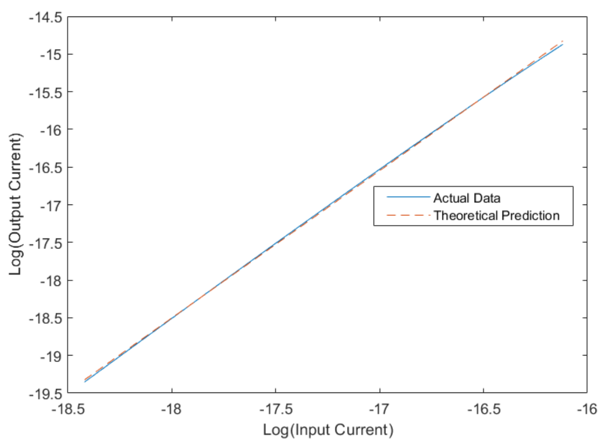


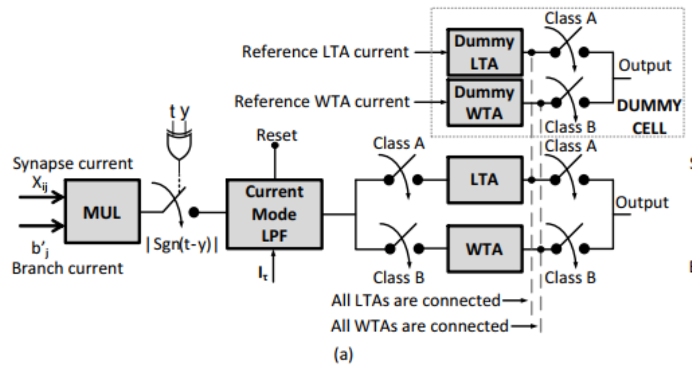
Fig. 6. Comparison to ideal behaviour for Squaring Circuit.

constitute the basic WTA circuit. All the input currents are compared and the maximum input current WTA takes most of the I_{BIAS} and wins, while the other output currents remain zero. M_{W13}, M_{W14} and M_{W15} convert the output currents to equivalent '0' or '1'. The output of the winner WTA becomes '1' while the others are '0'. The simulation results of the WTA circuit are shown in Fig. 9.

g) Active Synapse: Detection of worst performing synapse can provide faulty reading if one or more synapse inputs do not have an input pulse. Absence of spiking input will lead to noise current which would be erroneously selected by the LTAs upon comparison. To solve this problem, the concept of active synapses is introduced. An active synapse must have atleast one spike to be considered for comparison otherwise it will not be connected with the other LTAs. It is also included for WTAs as shown in Fig. 5(e) and Fig. 5(f).

h) Dummy LTA/WTA: One dummy cell in each of the copy synapse rows, silent synapse rows is incorporated to avoid faulty detection of worst and best performing synapse. Each dummy cell as shown in Fig. 7 include one LTA and one WTA in the coefficient calculator, but their input currents are fixed to a certain minimum reference value. These dummy LTAs and WTAs are connected to the normal coefficient calculator's WTAs and LTAs. During learning, initially the dummy LTA should always win ('1') and the dummy WTA should lose ('0') as the active synapses at least have 1 pulse at its input. These dummy LTAs and WTAs ensure that the output currents of the synapses are sufficiently high to be compared with each other and prevent any false triggering.

i) Class Comparator: As shown in Fig. 3, the squared current from each of the dendritic branches are summed together in each cell and compared to give the class of the input pattern. Here, the class comparisons are also carried out by the WTA circuit. Even here one dummy WTA is included to ensure the output current of each cell be more that this reference current to provide an output class.



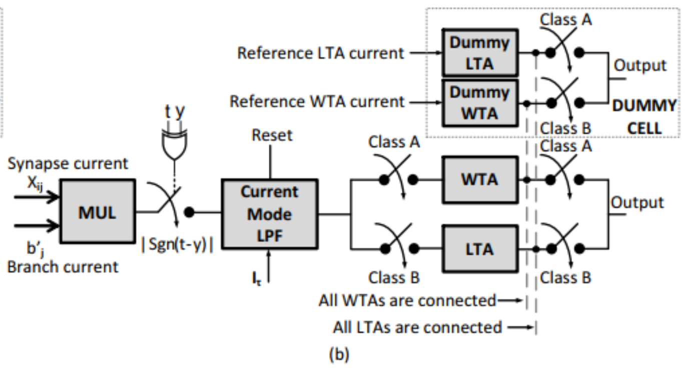


Fig. 7. Coefficient calculator: (a) for positive cell copy synapse and negative cell silent synapse, (b) for positive cell silent synapse and negative cell copy synapse.

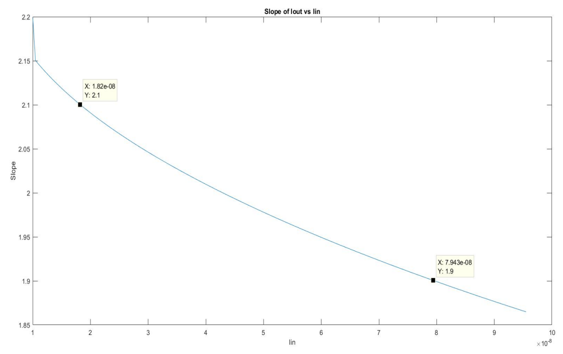


Fig. 8. Deviation from ideal behaviour in Current Multiplier.

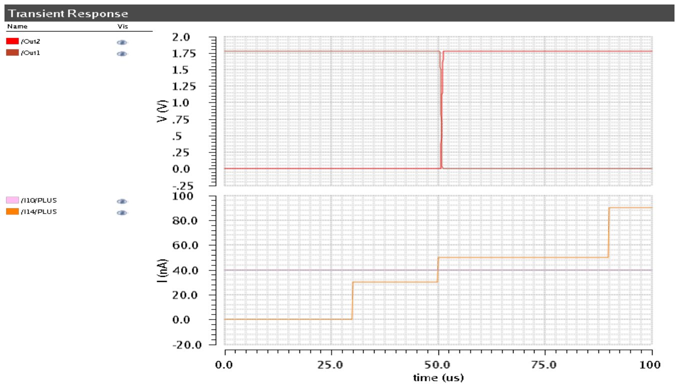


Fig. 9. SPICE simulation of WTA circuit.

j) c_{ij} Calculator: Coefficient calculator is the most important block in this on-chip implementation as the connections in the neuronal cells are changed based on the coefficients, a parameter to measure the performance index of the synapses. The main focus of these coefficient calculators is to find out the worst performing synapse among the copy synapses and the best performing synapses among the silent synapses. The c_{ij} calculator mainly includes a multiplier and a current mode LPF for averaging. Averaging after multiplication is needed to get the average coefficient value for a certain time duration, if possible for the total pattern set. The output current of the LPF is compared by the WTA and LTA blocks to find the best and worst performing synapses.

In our current mode implementation it is difficult to handle negative coefficients or currents. One possibility is to set a common mode current and add or subtract the output current from this common mode current for positive and negative classes respectively, but this will increase power consumption. We found out a different way to solve this problem. Learning has been divided into four cases as discussed below. If the pattern belongs to the same class as the neuronal cell, it is a *type A* pattern whereas if the pattern belongs to a different class than that of the neuronal cell, it is called *type B* pattern. For example, while classifying MNIST patterns, if a ‘0’ is fed to the neuronal cell for class ‘0’ it is a type A pattern while for neuronal cells for classes ‘1’,‘2’...‘9’, it is a type B pattern.

Case I: Positive cell, Class A patterns

According to the expression of c_{ij}, it can either be zero or positive. So either no current or positive current is the output of the LPF. For the copy synapses we need to find worst performing synapse, so LTA will be used for this. For silent synapses to find out best performing synapses WTAs will be used.

Case II: Positive cell, Class B patterns

According to the expression of c_{ij}, it can be either zero or negative. This means the loser with negative coefficients is the winner with equal positive coefficients, thus for copy synapses, we can use WTAs to find the minimum among negative currents . For silent synapses, to find out the best synapse we have to find out the maximum among negative currents which can be done by LTAs.

Case III: Negative cell, Class A patterns

According to the expression of c_{ij}, it can be either zero or negative. So it will be similar to case II, the worst performing copy synapse will be found out by WTAs and the best performing silent synapse by LTAs.

Case IV: Negative cell, Class B patterns

According to the expression of c_{ij}, it can be either zero or positive. So, it will be similar to case I, the worst performing copy synapse will be found out by LTAs and the best performing silent synapse by WTAs.

Block diagram of all the above cases is shown in Fig. 7 along with the dummy cell that was discussed above. Every c_{ij} calculator associated with each copy and silent synapses need both LTA and WTA which will be selected based on the incoming pattern to find out the worst and best performing synapses.

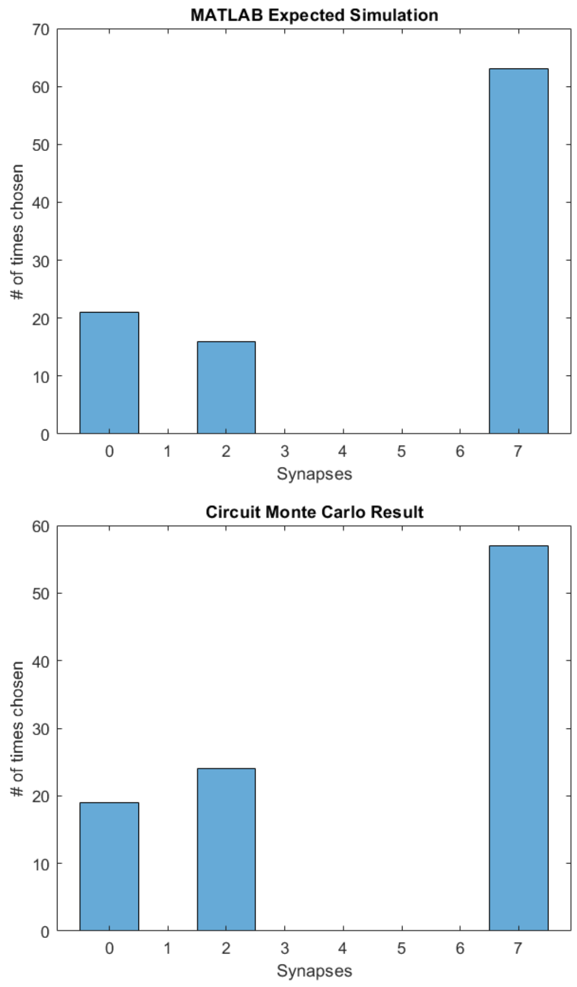


Fig. 10. Comparison of actual Monte Carlo simulations with the expected results.

VI. Monte Carlo Analysis

Statistical variations in VLSI devices can cause a degradation in classification accuracy if the correct synapses are not chosen during learning. To analyze the effects of statistical variation on the learning rule, Monte Carlo simulations were performed.

A sequence of 100 images was fed to a dendritic branch with 9 synapses. Due to the limited time constant of the LPF, c_{ij} values were calculated for each synapse by averaging over every 10 patterns. The lost information was made up for by voting. The synapse chosen most number of times by the WTA circuit, was the desired synapse. This process was repeated 100 times.

The Monte Carlo simulations of the branch revealed that synapse #7 was chosen $\approx 60\%$ of the times. Software simulations verified that synapse #7 should be chosen as the winner ideally. When the ideal c_{ij} values were multiplied with a noise ($\mu = 1$ and $\sigma = 0.2$), the plot obtained had a behaviour similar to that of the circuit.

VII. Conclusion

Main contribution of this work is to demonstrate feasibility of hardware implementation of the NRW learning algorithm in a neuromorphic architecture. The algorithmic modifications made to the NRW learning rule accelerated the rate of learning and made on-chip implementation feasible. An efficient architecture was proposed and the robustness of the design to variations was verified using Monte-Carlo simulations.