LETTER Communicated by Giacomo Indiveri

Spiking Neural Classifier with Lumped Dendritic Nonlinearity and Binary Synapses: A Current Mode VLSI Implementation and Analysis

Aritra Bhaduri
aritra002@e.ntu.edu.sg
Amitava Banerjee
amitavabanerjee.iitkgp@gmail.com
Subhrajit Roy
subhrajit.roy@ntu.edu.sg
Sougata Kar
sougatakar@gmail.com
Arindam Basu
arindam.basu@ntu.edu.sg
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

We present a neuromorphic current mode implementation of a spiking neural classifier with lumped square law dendritic nonlinearity. It has been shown previously in software simulations that such a system with binary synapses can be trained with structural plasticity algorithms to achieve comparable classification accuracy with fewer synaptic resources than conventional algorithms. We show that even in real analog systems with manufacturing imperfections (CV of 23.5% and 14.4% for dendritic branch gains and leaks respectively), this network is able to produce comparable results with fewer synaptic resources. The chip fabricated in $0.35\,\mu$m complementary metal oxide semiconductor has eight dendrites per cell and uses two opposing cells per class to cancel common-mode inputs. The chip can operate down to a $V_{dd} = 1.8$ V and dissipates 19 nW of static power per neuronal cell and $\approx$ 125 pJ/spike. For two-class classification problems of high-dimensional rate encoded binary patterns, the hardware achieves comparable performance as software implementation of the same with only about a 0.5% reduction in accuracy. On two UCI data sets, the IC integrated circuit has classification accuracy comparable to standard machine learners like support vector machines and extreme learning machines while using two to five times binary synapses. We also show that the system can operate on mean rate encoded spike patterns, as well as short bursts of spikes. To the best of our knowledge, this is the

A. Bhaduri and A. Banerjee contributed equally to this work.

Neural Computation **30, 723–760** (2018)
doi:10.1162/NECO_a_01045

first attempt in hardware to perform classification exploiting dendritic properties and binary synapses.

1 Introduction

Spiking neural networks (SNN), considered to be the third generation of neural networks, were proposed due to the advent of neurobiological evidence (Rieke, Warland, de Ruyter van Steveninck, & Bialek, 1999) that suggested biological neural systems use timing of action potentials, or spikes to convey information. These types of network are considered to be more biorealistic and computationally more powerful than their predecessors (Maass & Markram, 2004; Maass, 1999; Gutig & Sompolinsky, 2006). Neuromorphic engineering (Mead, 1990) aims to emulate the analog processing of neuronal structures in circuits to achieve low-power, low-area, very large-scale integrated circuit implementations. Thus, while theoretical studies on SNN have progressed rapidly, neuromorphic engineers have, in parallel, implemented low-power VLSI circuits that emulate sensory systems (Chan, Liu, & Schaik, 2007; Culurciello, Etienne-Cummings, & Boahen, 2003; Lichtsteiner, Posch, & Delbruck, 2008; Hsieh & Tang, 2012) and higher cognitive functions like learning and memory (Arthur & Boahen, 2007; Wang, Hamilton, Tapson, & van Schaik, 2014). Since silicon systems face many challenges similar to neuronal ones, we hope to gain insight into some operating principles of the brain by making such neuromorphic systems. Also, with the advent of brain-machine interfaces and the internet of things, there is now a pressing need for area- and energy-efficient neural networks for pattern classification.

Recently Roy, Basu, and Hussain (2013) and Hussain, Gopalakrishnan, Basu, and Liu (2013), proposed structures inspired by the nonlinear properties of dendrites in neurons that require many fewer synaptic resources than other neuromorphic designs. The learning of these structures involves network rewiring of binary synapses that is comparable to the structural plasticity observed in biological neural systems. As an example, in vivo imaging studies have shown that synaptic rewiring mediated by rerouting of whole axonal branches to different postsynaptic targets takes place in the mature cortex (Stettler, Yamahachi, Li, Denk, & Gilbert, 2006). Several experimental studies have provided evidence for the formation and elimination of synapses in the adult brain (Trachtenberg et al., 2002), including activity-dependent pruning of weaker synapses during early development (Be & Markram, 2006). Inspired by these phenomena, our learning algorithm tries to find the best sparse combinations of input on each dendrite to improve performance. This choice of connectivity can be easily incorporated in hardware systems using address event representation (AER) protocols, commonly used in current neuromorphic systems, where the connection matrix is stored in memory. Since this memory had to be stored

for implementing any AER-based system, no extra overhead is needed to implement our method other than the dendritic nonlinearity. Instead, the reduced number of synaptic connections translates to a reduction in memory access and communication overhead that is often the most power-consuming aspect of large-scale spiking neural networks (Hasler & Marr, 2013).

In this work, we present a neuromorphic current mode implementation of the above neural classifier with shared synapses driven by AER. Some of the initial characterization results were presented in Banerjee, Kar, Roy, Bhaduri, and Basu (2015). In this letter, we present complete characterization results and detailed results on pattern classification using the chip. The organization of the letter is as follows. We first present some architectural modifications of the basic dendritic cell for improved hardware performance. Next, we present circuit descriptions and simulations of each building block. Measurement results from a chip fabricated in 0.35 μm complementary metal oxide semiconductor (CMOS) are presented in the following section to prove functional correctness. Finally, we conclude with discussions in the last section.

2 Background and Theory

Roy and colleagues (Roy et al., 2013; Roy, Banerjee, & Basu, 2014) and Hussain and colleagues (Hussain, Gopalakrishnan, Basu, & Liu, 2013; Hussain, Basu, & Liu, 2014) have described spike train classifiers employing neurons with nonlinear dendrites (NNLD) and binary synapses. Due to the presence of binary synapses, the learning in these types of architectures happens by morphological changes in the connections between inputs and dendrites, not by weight update. Thus, these architectures are amenable to neuromorphic implementation employing AER protocols. In this letter, we present a circuit to implement the architecture proposed in Hussain et al. (2014) that has comparable performance to other spike-based classifiers such as O'Connor, Neil, Liu, Delbruck, & Pfeiffer (2013) but use 2 to 12 times fewer synaptic resources. Hence, our hardware implementation requires correspondingly less memory to store the connectivity information. It also needs proportionately less energy to communicate the connection information for each spike. Note that in this work, the training is done on a PC and the learned connection matrix is downloaded to the hardware platform for testing. Next, for completeness, we briefly describe the architecture of a basic NNLD, a classifier composed of two such NNLDs, the learning rule to train the classifier, and some modifications to improve hardware performance.

2.1 Architecture. Figure 1a shows the basic architecture of a single NNLD with m dendritic branches and with k excitatory synapses per branch as described in Hussain et al. (2013) and Roy et al. (2013). If a $d-$dimensional

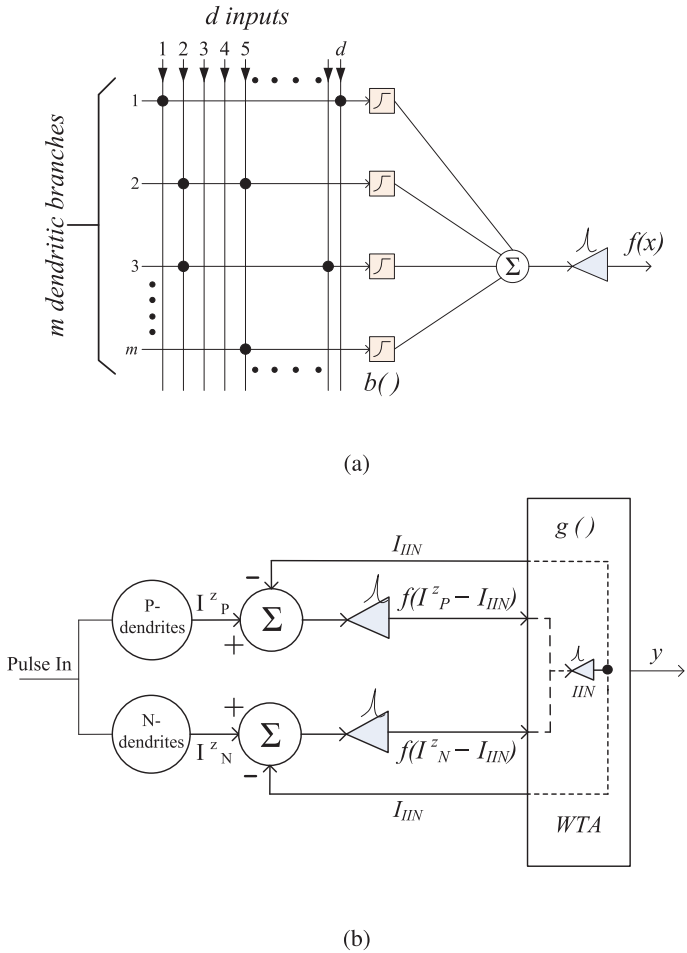


Figure 1: (a) Architecture of a single neuron with multiple dendritic branches and sparse synaptic connectivity. The black circles denote synaptic connections, and the $b()$-blocks represent lumped dendritic nonlinearities. $f()$ is the neuron activation function. (b) Single-ended classifier composed of two neurons with nonlinear dendrites (NNLD) and winner-take-all (WTA) for comparing the outputs of the neurons.

binary input pattern $\mathbf{x}$ ($d \gg k$) is applied to this system ($\mathbf{x} \in \{0,1\}^d$), then each synapse is excited by the input connected to it and the output response of the jth dendritic branch is given by $z_j = b(\sum_{i=1}^{k} w_{ij}x_{ij})$. Here $b()$ is a model of the dendritic nonlinearity given by $b(q) = q^2/x_{thr}$, w_{ij} is the synaptic weight of the ith synapse on the jth branch, x_{ij} is the corresponding

input, and x_{thr} is a scaling constant. We choose a square law nonlinearity since it has been shown in Hussain et al. (2015) to match the measured dendritic nonlinearity reported in Polsky, Mel, and Schiller (2004). Other popular nonlinearities like RELU will not be applicable here since the input to the nonlinearity is always positive because the synapses are only excitatory. Also, squaring circuits can be made using only five transistors, as we will show. Let $I^z = \sum_{j=1}^{m} z_j$ denote the total summed output current from all the m dendrites that enter the neuron Then the overall output $f(I^z)$ of a single neuronal cell is given by

$$f(I^z) = f\left(\sum_{j=1}^{m} b\left(\sum_{i=1}^{k} w_{ij}x_{ij}\right)\right) (f : \Re \to \Re) \tag{2.1}$$

$$= K_{neu} \times \left(\sum_{j=1}^{m} b\left(\sum_{i=1}^{k} w_{ij} \times x_{ij}\right)\right) \tag{2.2}$$

where $f(z) = K_{neu} \times z \times u(z)$ denotes the linear neuronal current-frequency conversion function where $u()$ is the Heaviside function defined as $u(n) = 1$ for $n > 0$, $u(n) = 0.5$ for $n = 0$, and $u(n) = 0$ for $n < 0$. This signifies that the neuron produces zero outputs for negative inputs.

For rate-encoded spike trains at the input, a low-pass filtering synaptic kernel $K(t)$ is used in the earlier work (Hussain et al., 2013; Hussain, Basu, & Liu, 2014; Roy, Banerjee et al., 2014) to produce an average synaptic output current $\overline{I_{syn}}$ proportional to the input spike rate. This average current for the ith synapse on the jth branch is denoted as x_{ij} in equations 2.1 and 2.2. Hussain et al. (2014) constructed a two-class classifier by comparing the output of two NNLDs (denoted by P and N) using winner-take-all (WTA) based on an inhibitory interneuron (IIN; Oster, Douglas, & Liu, 2009). This structure is similar to the canonical microcircuit in Douglas, Martin, and Whitteridge (1989) but without the recurrent excitatory connections. This simple classifier is shown in Figure 1b where $g : \Re \times \Re \to \{0, 0.5, 1\}$ denotes the operation of the WTA. I_P^z and I_N^z denote the total currents from the dendritic branches of P- and N-cells, respectively. The operation of the WTA is as follows. The IIN produces a current I_{IIN} proportional to the firing rates of the two excitatory neurons. This current provides inhibition to both input currents and causes spike frequency adaptation in the excitatory neurons. We can express the current I_{IIN} as

$$I_{IIN} = K_{IIN}(f(I_P^z - I_{IIN}) + f(I_N^z - I_{IIN})) \tag{2.3}$$

where K_{IIN} denotes the effective gain from the input spikes to output current of the IIN. Suppose $I_P^z > I_N^z$. In that case, it can be shown that $I_P^z > I_{IIN} > I_N^z$ if $\frac{K_{IIN}K_{neu}}{1+K_{IIN}K_{neu}} I_P^z > I_N^z$, a mild condition that is almost always true if $K_{IIN} \gg 1$.

Thus, due to the inhibitory current, the weaker of the two inputs gets cancelled, and the stronger persists at a diminished value. This helps in reducing the firing rates of the excitatory neurons. We will refer to this architecture as a single-ended classifier. Since logically the WTA is selecting one of the inputs as the winner (larger), we can express the output y of the single-ended classifier as

$$y = g(f(I_P^z - I_{IIN}), f(I_N^z - I_{IIN}) \tag{2.4}$$

$$= u(f(I_P^z - I_{IIN}) - f(I_N^z - I_{IIN})) = u(K_{neu}(I_P^z - I_N^z)), \tag{2.5}$$

where the function $g()$ is a Heaviside function of the difference of inputs. The last step shows that we can identify the winner even by just noting I_P^z and I_N^z or, effectively, $f(I_P^z)$ and $f(I_N^z)$. This is the approach we take in our prototype IC and do not include the IIN of the WTA. This serves as a proof of concept for classification accuracy with the understanding that the high firing rates in measurements will be reduced in future versions with inclusion of on-chip WTA.

2.2 Learning Algorithm: Network Rewiring Rule. The classifier shown in Figure 1b was trained by a structural plasticity-based network rewiring learning rule. For each input pattern, a binary target signal $t \in \{0, 1\}$ is provided to the classifier. The training was based on mean rate binary inputs $\mathbf{x} \in \{0, 1\}^d$, and testing was performed by mapping each input dimension to a Poisson spike train with high or low mean firing rates. The learning algorithm primarily consists of the following steps:

1. The inputs are connected to the dendritic branches of the NNLDs via binary synapses ($w_{ij} = 0$ or 1), so the network learns through connection changes instead of weight changes. Since $d \gg k$, learning involves choosing the best k choices of connection for each dendrite.
2. At each epoch of learning, a randomly selected set T of n_T synapses is chosen for possible replacement. The performance index c_{ij} for each synapse in the set n_T is computed as $c_{ij} = \langle x_{ij} b'_j (t - y) \rangle$ for the positive cell and $c_{ij} = -\langle x_{ij} b'_j (t - y) \rangle$ for the negative cell. Here, $\langle\ \rangle$ denotes averaging over the entire training set. The synapse having the least value of c_{ij} in T ($T_{\min}$) is tagged for replacement.
3. For replacement, a candidate set R is formed by randomly selecting n_R of the d input lines. The synapses in R are placed on the dendrite with the lowest c_{ij} synapse from the previous step. The performance index c_{ij} is again computed for synapses in R, and the synapse having the highest c_{ij} ($R_{\max}$) is used to replace $T_{\min}$.

For more details about the algorithm, refer to Hussain, Liu, and Basu (2015) and Roy, Banerjee et al. (2014). We also invite readers to look into Roy et al.

(2013) and Hussain et al. (2014). For a detailed description of the architectures and learning rules, see Roy et al. (2013) and Hussain et al. (2014).

2.3 Modifications for Hardware Implementation. Three modifications of the single-ended classifier and learning algorithm are used to improve performance of the hardware.

2.3.1 Differential Architecture. We found that hardware implementation of the single-ended architecture of Figure 1b has the problem of high baseline or common-mode current at the input of the spiking neurons, leading to frequency saturation whenever the number of dendrites m or firing rates of input afferents denoted by x_{ij} grow large. In other words, due to a large number of dendrites producing positive currents, even arbitrary connections before learning will generate a large amount of current that can saturate the neuron. Hence, we modified the architecture to a differential one, reported in Hussain et al. (2015) and shown in Figure 2a with the following equation:

$$y = g(f(I_P^z - I_N^z - I_{IIN}),\ f(I_N^z - I_P^z - I_{IIN})) \tag{2.6}$$

$$= u(f(I_P^z - I_N^z - I_{IIN}) - f(I_N^z - I_P^z - I_{IIN})) = u(K_{neu}(I_P^z - I_N^z)). \tag{2.7}$$

Although the final expression in equation 2.7 is same as the one in equation 2.5, the modification in the intermediate step makes the performance more robust to architectural changes like changing m (the number of dendrites per cell) since the system performance is now less dependent on change in average currents (average quantities get cancelled when evaluating $(I_P^z - I_N^z)$ or $(I_N^z - I_P^z)$). It should again be noted that the final winner can be inferred without having the IIN on chip, albeit at the cost of producing more spikes. However, this was the preferred option for simplicity in the test chip.

2.3.2 Dendritic Leak. The second modification to the basic architecture is shown in Figure 2b: a constant dendritic leak current is subtracted from the total synaptic input current on the branch to form the input to the dendritic nonlinearity. In other words, we can modify the equation of the dendritic nonlinearity to become a new function $b_{leak}()$ given by

$$b_{leak}(q) = \frac{(q - q_{leak})^2}{x_{thr}} u(q - q_{leak}), \tag{2.8}$$

where q_{leak} is the dendritic leak current and $u()$ denotes the Heaviside function. As Hussain et al. (2013) showed, large values of k (synapses per dendrite) led to saturation of each dendritic nonlinearity, reducing performance. In the hardware implementation, even without having saturation in each branch, large values of k lead to a high average current per

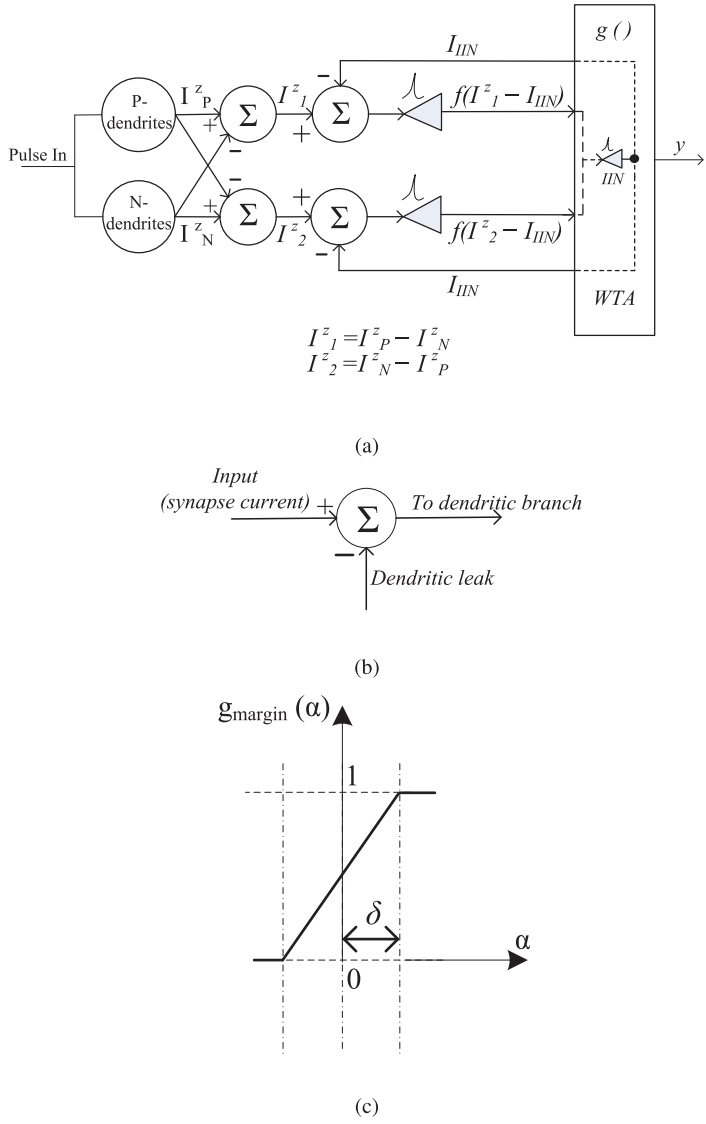


Figure 2: Three modifications to the basic NNLD-based classifier for improved performance in hardware. (a) Modification of the single-ended architecture to a differential one for increased robustness to architectural changes and less dependence on average currents. (b) Subtraction of constant dendritic leak from total synaptic input current to reduce the input current to dendritic nonlinearity. (c) Graphical representation of the $g_{margin}()$ function used during training to achieve better generalization.

branch, increasing power dissipation after squaring tremendously. Also, since we employed subthreshold translinear design principles to implement the square nonlinearity, large input currents cause the transistors to enter above the threshold regime, causing deviations from ideal behavior. Hence, adding dendritic leak q_{leak} helps in removing common-mode current in every branch and helps to increase the range of usable k values.

2.3.3 Margin-Based Learning. The last modification shown in Figure 2c is a change in the learning procedure and does not modify the architecture. As Hussain et al. (2014) showed, using a function $g_{margin}()$ for the WTA, as shown in Figure 2c, instead of the Heaviside $g()$ during training allows building a margin of δ around the classification boundary. In other words, the output y during training is given by

$$y = g_{margin}(K_{neu} \times (I_P^z - I_N^z)). \tag{2.9}$$

During testing, the normal Heaviside $g()$ is used as before. This helps in increased generalization and noise robustness because the learning algorithm does not stop modifying connections for a pattern immediately when it is correctly classified. As an example, consider a pattern for which $t = 1$ but $y = 0$ because $I_P^z < I_N^z$. In such a case, the learning algorithm tries to modify connections to increase I_P^z and decrease I_N^z for this pattern. If $y = g(K_{neu} \times (I_P^z - I_N^z))$ is used, the error for this pattern becomes zero immediately when $I_P^z > I_N^z$ since $g()$ is a Heaviside function. Hence, it is quite possible for inputs slightly different from those encountered during training to be wrongly classified due to slight changes in I_P^z or I_N^z. However, using the $g_{margin()}$ function shown in Figure 2c, the error $t - y$ during training is nonzero until $K_{neu} \times (I_P^z - I_N^z) > \delta$. Intuitively, as long as the effect of noise does not alter any of the currents I_P^z or I_N^z by magnitudes approximately larger than δ/K_{neu}, noisy inputs during testing will not lead to misclassifications. The detailed algorithm to choose and adapt δ is given in Hussain et al. (2015). We show the effect of adding margin during training in our measurement results in section 5. Table 1 summarizes the notations introduced in this section that will be used throughout the remainder of the letter.

3 VLSI Implementation of Neuromorphic IC

The VLSI architecture of the implemented neuromorphic integrated circuit (IC) is shown in Figures 3a to 3d, where AER is used to provide the synaptic input. A differential pair integrator (DPI) circuit has been used to implement the synaptic function of the neuron; in this circuit, it is possible to achieve the linear filtering property of each synapse (Bartolozzi & Indiveri, 2007) by proper biasing. This linear filtering property is implemented here to replace

Table 1: Notations Used in This Letter.

$\mathbf{x}$: Input vector
d: Dimensionality of the input
m: Number of dendrites per cell
k: Number of synapses per dendrite
$b()$: Nonlinear dendritic function without leak
$b_{leak}()$: Nonlinear dendritic function with leak
x_{ij}: Input value at the ith synapse of the jth dendrite
x_{thr}: Scaling constant of the nonlinear dendritic function
$g()$: Input-output function of WTA without margin
$g_{margin}()$: Input-output function of WTA with margin
I_P^z: Sum of the currents from dendritic branches of P cell
I_N^z: Sum of the currents from dendritic branches of N cell
δ: Margin of classification
f_{HIGH}: Mean rate of Poisson spike train mapped to binary input 1
f_{LOW}: Mean rate of Poisson spike train mapped to binary input 0

all the synapses in one dendritic connection by a single shared synapse, drastically reducing the effective layout area of the IC.

This IC is interfaced with a field programmable gate array (FPGA) controller that generates input spikes and addresses, as shown in Figure 4. The learned sparse connectivity matrix is stored inside the FPGA memory in a very compressed form by using two look-up tables. This constitutes the crossbar or routing array shown in Figure 1a. Based on the input line address, the controller reads the connectivity information of the address line from memory and generates an n-bit decoder address to route the spike to the proper dendrite. Spikes from the FPGA output reach the synapse circuit input through an n-bit to a 2^n-bit address decoder followed by digital switches. There are, in all, $2^{(n-1)}$ P-dendrites and $2^{(n-1)}$ N-dendrites connected to a NEURON block. The shared synapse circuit in each dendrite is followed by a square-law nonlinear circuit. For the chip that we have fabricated and present in section 5, $n = 4$ implies that both P- and N-cells have $m = 8$ dendrites. The difference between the total currents from the $2^{(n-1)}$ P-dendrites and the $2^{(n-1)}$ N-dendrites appears to the input of the neuron block and is converted to equivalent output spikes.

The output of the neuron block is digitized for proper handshaking of the IC with the FPGA in the form of an *REQ* and an *ACK* signal. We have implemented only one cell to compute one term ($f(I_P^z - I_N^z)$) in equation 2.6. (Unless otherwise mentioned, the inhibitory interneuron is not to be taken into account.) The second term, ($f(I_N^z - I_P^z)$), is computed by passing the same input again through the IC, but with the connection matrix of the P-cell interchanged with that of the N-cell. In this case, the output of the IC is $f(I_N^z - I_P^z)$. This is shown in Figure 3e. So basically, the two units presented in Figure 2c are implemented through passing the same input spike train

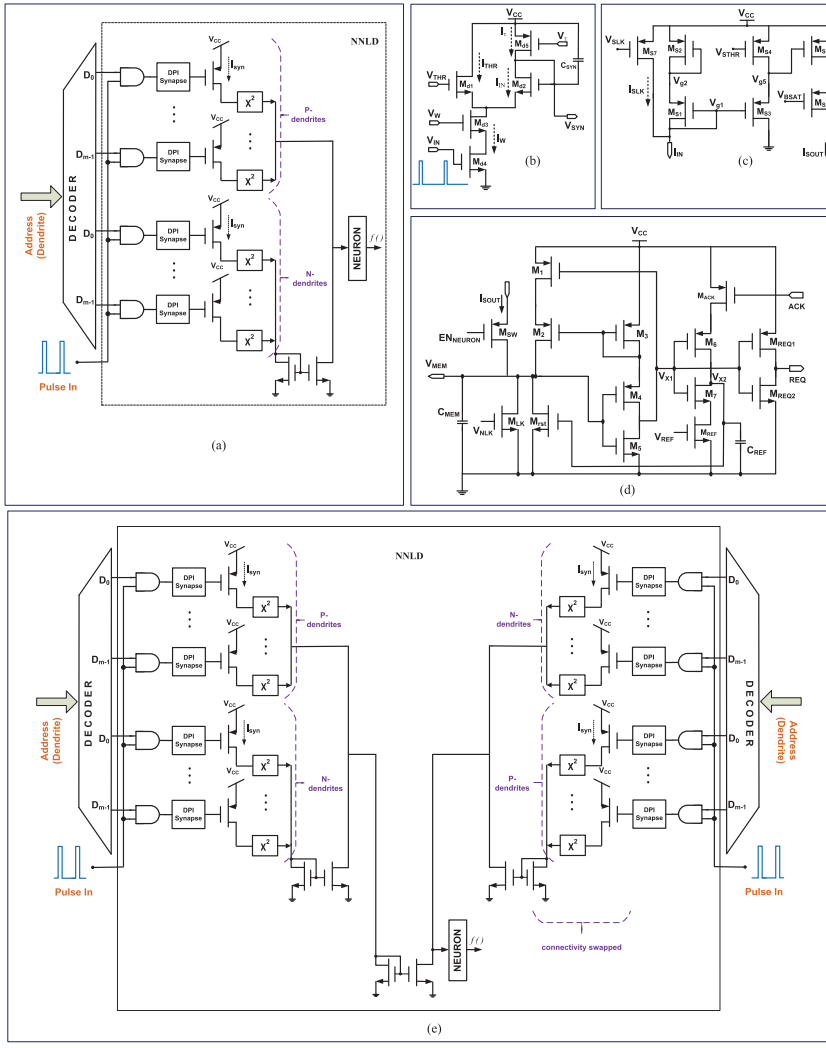


Figure 3: (a) VLSI architecture of the implemented neuromorphic IC composed of $m = 8$ dendritic branches with shared DPI synapses in P- and N-cells (from Banerjee et al., 2015). Each dendrite has a squaring nonlinearity, and the difference of the currents from P and N dendrites enters an integrate-and-fire (I&F) neuron. Detailed circuit diagrams from Banerjee et al. (2015) of subblocks (b) DPI synapse, (c) squaring block employing translinear principle, and (d) I&F neuron. (e) Generation of the difference currents $(I_P^z - I_N^z)$ and $(I_N^z - I_P^z)$ by swapping the excitatory and inhibitory connections for the differential architecture.

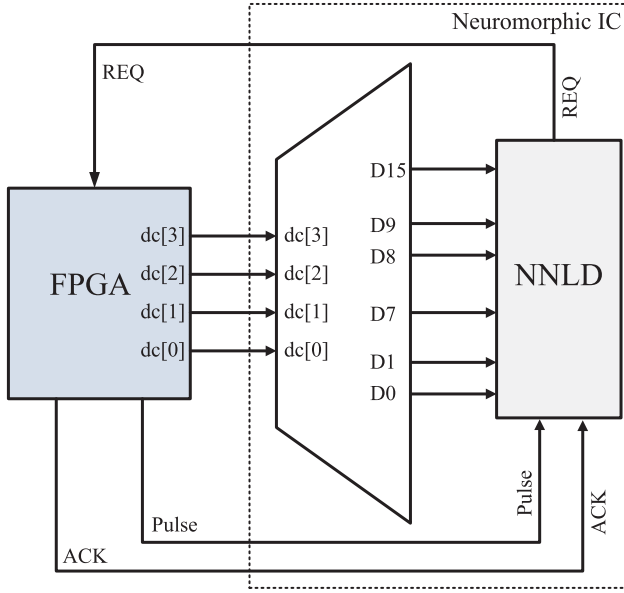
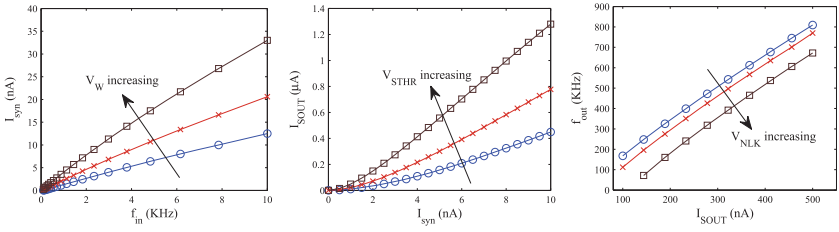


Figure 4: FPGA-IC interface where the FPGA supplies input spikes to individual dendrites of the IC and captures output spikes from the neuron (from Banerjee et al., 2015).

into the same circuit but with the swapped connectivity. The final decision of the pattern class is taken on the PC after computing the difference of these two results and applying the Heaviside function to it. Thus, although the chip does not itself perform learning, most of the processing from input to output is done on-chip.

The neuromorphic IC has been fabricated in AMS 0.35 μm CMOS technology, and each classifier comprising two neurons occupies $820 \times 320\ \mu m^2$ area. In this section, we briefly describe the circuit implementation of these blocks and show simulation results to describe the functionality of each block.

3.1 Synapse Circuit. The circuit schematic shown in Figure 3b is a differential paired integrator (DPI) circuit that converts presynaptic voltage pulses into postsynaptic currents, $I_{syn}(t)$. In this circuit, transistors operate in the subthreshold regime. As Bartolozzi and Indiveri (2007) mentioned, the bias voltages V_w and V_τ are used to set $I_w \gg I_\tau$, which simplifies this nonlinear circuit to a canonical first-order low-pass filter. For an input spike (arriving at t_i^- and ending at t_i^+) applied to the DPI synapse, the rise time of I_{syn} is very small. The discharge profile can be modeled by the following equation.



(a) (b) (c)

Figure 5: SPICE simulation results of (a) a synapse block showing the linearity of average current with input frequency ($V_w = 0.48$ V, 0.50 V, 0.52 V), (b) a square nonlinearity block ($V_{STHR} = 2.87$ V, 2.90 V, 2.93 V), and (c) a neuron block showing output frequency versus input current relationship ($V_{NLK} = 0.37$ V, 0.40 V, 0.43 V).

$$I_{syn}(t) = I_0 e^{\left(-\frac{t-t_i^+}{\tau_s}\right)}, \tag{3.1}$$

where $\tau_s = \frac{C_{SYN} U_T}{\kappa I_\tau}$, κ is the subthreshold slope factor and U_T is the thermal voltage. This synapse circuit, when typically excited with a continuous pulse train of average frequency f_{in} and pulse duration Δt, will have a steady-state output current $\overline{I_{syn}}$ (average of the output transient current) that holds a linear relationship with f_{in}. The relationship has been derived in Bartolozzi and Indiveri (2007) as

$$\overline{I_{syn}} = K'_{syn} \times (\Delta t) \times f_{in} = K_{syn} \times f_{in}, \tag{3.2}$$

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 for this block, shown in Figure 5a, demonstrate the expected linearity. Also, for larger values of weight governed by the bias V_w, the slope of this curve increases as expected. The maximum output current of this circuit is limited by the constraint of keeping the transistor driven by V_{syn} and creating the current I_{syn} (see Figure 3a) in subthreshold. Denoting the threshold current of the MOS creating I_{syn} by $I_{thr,mos}$, this constraint can be written as

$$\overline{I_{syn}} \leq I_{thr,mos} \Longrightarrow f_{in} \leq \frac{I_{thr,mos}}{K_{syn}}. \tag{3.3}$$

3.2 Square Block Circuit. We have designed the current mode squaring circuit given in Figure 3c as described in Hussain et al. (2013). The transistors M_{S2}, M_{S1}, M_{S3}, and M_{S5} form a translinear loop. To implement the dendritic leak, the transistor M_{S7} is added with a current I_{SLK} set by the bias

V_{SLK}. This current gets subtracted from the input current $I_{in} = \overline{I_{syn}}$ coming from the synapse. Hence, the current through M_{S5} can be expressed as given by

$$I_{SOUT} = \frac{(I_{in} - I_{SLK})^2}{I_{STHR}} = \frac{(\overline{I_{syn}} - I_{SLK})^2}{I_{STHR}}. \tag{3.4}$$

The transistor M_{S5} is biased to pass a maximum current of I_{sat} (set by V_{BSAT}) that can implement a saturating nonlinearity if needed. I_{STHR} is the DC current through M_{S4} set by its gate voltage (V_{STHR}). Figure 5b shows simulation results of the square nonlinearity block when $I_{SLK} \approx 0$. Also, it can be seen that increasing V_{STHR} reduces I_{STHR} in equation 3.4 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 subthreshold. Assuming $I_{SLK} \approx 0$ and all transistors in the translinear loop have the same dimensions with threshold current $I_{th,mos}$, the constraint on average synaptic current entering this block can be written as

$$\overline{I_{syn}} \leq \sqrt{I_{STHR,\max} I_{SOUT,\max}} = I_{thr,mos}. \tag{3.5}$$

It can be seen that the constraint set by equation 3.5 is the same as the one set by equation 3.3 if I_{STHR} can be set to its maximum value of $I_{thr,mos}$.

3.3 Neuron Circuit. The circuit diagram of the implemented spiking neuron as described in Indiveri (2003) is depicted in Figure 3d. It comprises one integrating capacitor C_{MEM}, an inverter (M_4 and M_5) with positive feedback circuit (M_1, M_2 and M_3), transistors M_{REF} and M_{rst} to control the refractory period of the neuron, a leakage transistor (M_{LK}) for controlling the minimum I_{OUT} to start charging C_{MEM}, M_{REQ1}, and M_{REQ2} to generate REQ signal to the FPGA, and M_{ACK} to be turned on for an ACK from FPGA. The output current of the square block (I_{SOUT}) is integrated by C_{MEM} to generate the dynamics of the membrane voltage (V_{MEM}). As V_{MEM} increases and approaches the switching voltage of the inverter (M_4-M_5), the feedback current starts flowing through M_1-M_2 causing a sudden rapid increase of the V_{MEM} profile and generates an REQ signal indicating spike emission. The positive feedback reduces the short circuit current in the inverter, helping to reduce power dissipation. When an ACK signal is received, C_{MEM} is quickly discharged back to ground through the reset transistor M_{rst}, the on-resistance of which is controlled by V_{X2}. When C_{MEM} is fully discharged, V_{X1} is driven back to V_{CC}, causing M_7 to turn on, which slowly discharges the node V_{X2} through transistor M_{ref}. As long as V_{X2} is sufficiently high, M_{rst} is active and is clamped to ground. This is called the refractory period of the neuron; hence, there will be no REQ pulse generated by the neuron during this period. The entire current at the output of the square block will

pass through M_{rst}. The input-output characteristic (for a negligible refractory period) can be represented by equation as

$$f_{out} = K_{neu} \times (I_{SOUT} - I_{NLK}), \tag{3.6}$$

where f_{out} denotes the spiking frequency of the neuron. I_{NLK} is the leakage current through M_{LK} set by V_{NLK}; hence, $I_{SOUT} - I_{NLK}$ is the total current charging the neuron membrane capacitance. Finally, K_{neu} denotes the slope of the i-f curve of the neuron. Figure 5c displays the above linear relationship between f_{out} and I_{SOUT} for the negligible refractory period. Also, the effect of increasing the leak current that causes a rightward shift of the $f - i$ curve has been shown in Figure 5.

Finally, combining equations 3.2, 3.4, and 3.6, we get the nonlinear $f_{in} - f_{out}$ relationship of the IC given as

$$f_{out} = K_{neu} \left[\frac{(K_{syn} \times f_{in} - I_{SLK})^2}{I_{STHR}} - I_{NLK} \right]. \tag{3.7}$$

4 FPGA Controller

To validate the operation of the neuromorphic IC, an FPGA-based controller logic (using Opal Kelly XEM 3010) has been implemented for generating spiking events to emulate the real-time behavior of a spike-based sensor. The address event representation (AER) protocol (Boahen, 2000), which is commonly used for other asynchronous neuromorphic hardware (Chan et al., 2007; Brink et al., 2013); is used for communication between the IC and the FPGA controller. The FPGA controller shown in Figure 6 further performs the task of using these input pulse addresses to determine a corresponding decoder or dendritic address that needs to be pulsed. Details of the controller are presented in the appendix.

5 Measurement Results

We have designed the dendritic classifier IC in the AMS 0.35 μm process and evaluated performance with the test setup described in the section 4. The microphotograph of the fabricated chip is shown in Figure 7. The DPI synapse, the square block, and the neuron combined together form a basic unit of the fabricated chip governed by equation 3.7. For this chip, each P- and N-cells has $m = 8$ dendrites. We first show some characterization results of the chip followed by pattern classification under different conditions.

5.1 Characterization. The input to the chip can be given only in terms of spikes or pulses, and only the output neuronal spike frequency can be

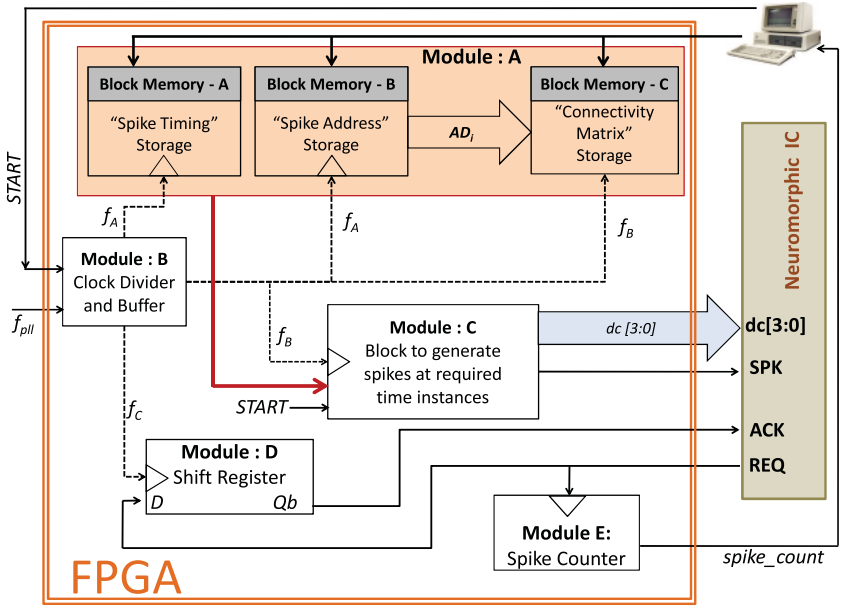


Figure 6: Internal architecture of the controller blocks implemented on FPGA for interfacing with the neuromorphic IC. Module A stores the spike times and addresses of input, as well as the connectivity matrix. Module C creates input spikes for the IC, and modules D and E receive events from the IC.

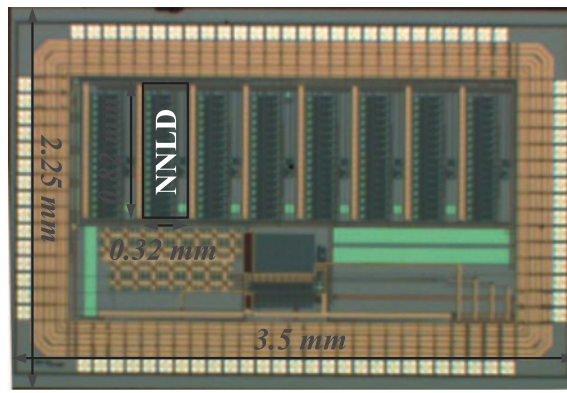


Figure 7: Microphotograph of the neuromorphic IC.

measured. Hence, we characterized the functionality of the chip in terms of $f_{in} - f_{out}$ curves. In other words, each of the subblocks—synapse, square nonlinearity, and neuron—cannot be separately characterized, but their

relative contribution to the $f_{in} - f_{out}$ curve can be ascertained by changing the biases associated with them. This method was used to first ensure the full functionality of each block. Note that periodic spike trains were used for characterization. Next, it was found that there is some leakage current–induced spurious firing of the neuron even without input pulses. The neuronal leak was adjusted (by regulating or tuning the corresponding bias voltage) until this spurious firing disappeared (so that $I_{NLK} = 0$) and the $f_{in} - f_{out}$ curve started from $f_{in} = 0$ for zero dendritic leak. In that case, we can rewrite equation 3.7 as

$$f_{out} = K_{neu} \frac{(K_{syn} \times f_{in} - I_{SLK})^2}{I_{STHR}} \times u(K_{syn} \times f_{in} - I_{SLK}) \tag{5.1}$$

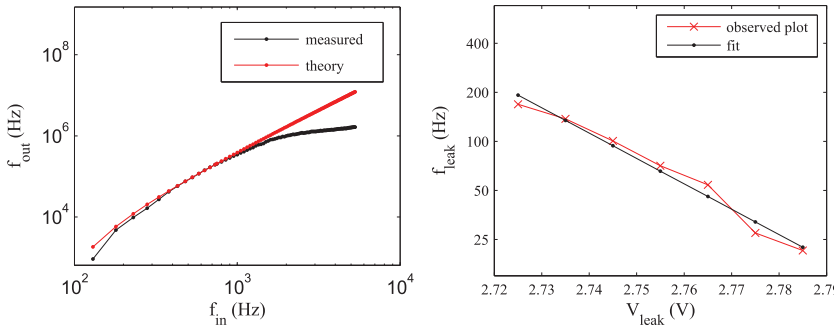
$$= K_{tot} \times (f_{in} - f_{leak})^2 \times u(f_{in} - f_{leak}), \tag{5.2}$$

where $K_{tot} = K_{neu} \times K_{syn}^2 / I_{STHR}$, $f_{leak} = I_{SLK}/K_{syn}$, and $u()$ is the Heaviside function. This implies that for zero neuronal leak, the neuron block does not fire at its output until and unless the steady-state current from the synapse block due to the spiking input exceeds the dendritic leak current. Further, assuming subthreshold operation of the transistor M_{S7}, the current $I_{SLK} = I_{SLK0} e^{\Delta V_{leak}/U_T}$ where I_{SLK0} is the value of I_{SLK} when the bias voltage $V_{leak} = V_{leak0}$ and ΔV_{leak} is defined as $\Delta V_{leak} = V_{leak} - V_{leak0}$. So equation 5.2 can be rewritten as

$$f_{out} = K_{tot} \times (f_{in} - f_{leak})^2 = K_{tot} \times (f_{in} - f_{leak0} e^{\Delta V_{leak}/U_T})^2, \tag{5.3}$$

where f_{leak0} is the value of f_{leak} when $V_{leak} = V_{leak0}$. Figure 8a plots the measured $f_{in} - f_{out}$ curve for one of the dendritic branches with $V_{leak} = V_{leak0}$ along with a fit using equation 5.3. Note that the y-axis is on a logarithmic scale. The curve departs from the expected square law at low values of f_{in} when the effect of f_{leak} becomes prominent. It also deviates at high values of f_{in} (larger than about 1.2 kHz) due to transistors in the synapse and the square blocks entering the above-threshold regime. For the same branch, the value of V_{leak} was then varied, and f_{leak} values were obtained for each case. These values are plotted in Figure 8b and fitted to the theoretically expected $f_{leak} = f_{leak0} e^{\Delta V_{leak}/U_T}$ as shown. This model was next used to set desired bias conditions during future classification experiments. Of course, there is some mismatch between each branch that can be characterized in terms of mismatch between K_{tot} and f_{leak0} for each branch.

This is shown in Figure 9. It should be noted that the two parameters K_{tot} and f_{leak0} together represent the total mismatch from each branch, and the effect of this mismatch cannot be eliminated by multiplexing the same neuromorphic circuits, as we have done, to get the classifier output. There is a significant mismatch across the branches with CV of 23.5% and 14.4% in K_{tot} and f_{leak0}, respectively. We will show later how the margin enhancement



(a) (b)

Figure 8: (a) Measured $f_{in} - f_{out}$ characteristics of one dendritic branch with corresponding theoretical fit. Note the logarithmic scale on both axes. (b) Extraction of parameter f_{leak0} for one dendritic branch. Note the logarithmic scale on the y-axis.

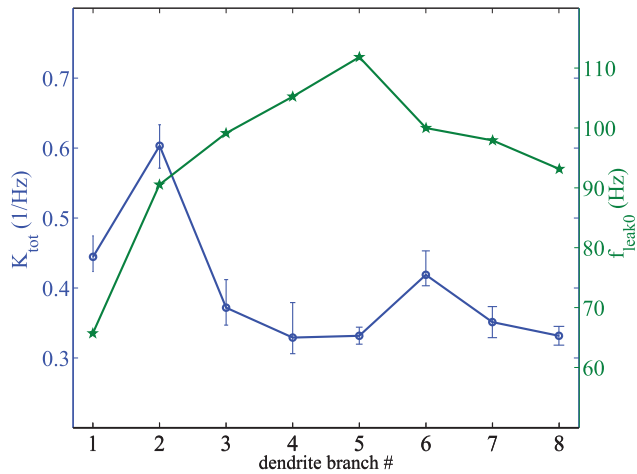


Figure 9: Variation of K_{tot} (values averaged over four trials) and f_{leak0} for the eight dendrites of the P-cell. The colors indicate the vertical axes corresponding to the plots.

algorithm helps achieve classification accuracies close to those of software. In the future, these separate values for each branch will also be used in the learning algorithm to calibrate for the mismatch.

Next, we characterized the power dissipation of the chip for different V_{dd} values. The chip is functional for power supply voltages as low as 1.8 V

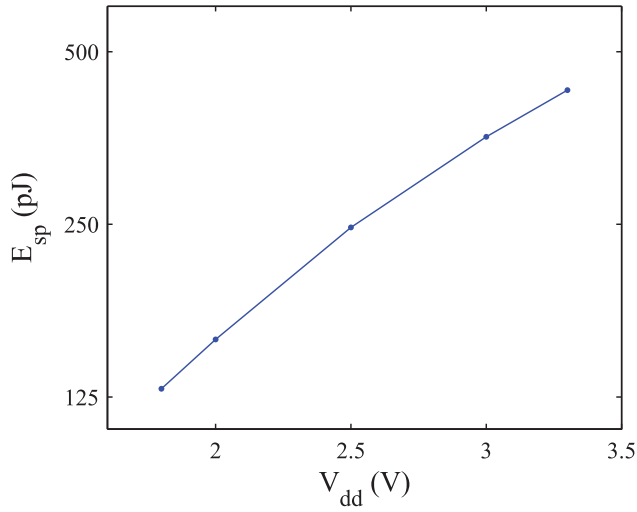


Figure 10: Energy dissipated per spike by the IC as a function of the power supply V_{dd}. Note the logarithmic scale on the y-axis.

due to the current mode design. The static current is approximately 10.5 nA, and the dynamic current depends on V_{dd}. The dynamic power is normalized to the spike frequency for different values of V_{dd} to get energy per spike (E_{sp}) and is plotted in Figure 10. The lowest value of $E_{sp} \approx 125$ pJ is attained at $V_{dd} = 1.8$ V. This characterization can be used to estimate the energy/classification operation. We expect this value to decrease quadratically when V_{dd} is reduced; hence, moving to a smaller process node like 65 nm and reducing V_{dd} to 0.45 V should reduce E_{sp} to approximately 8 pJ.

5.2 Pattern Classification: Random Binary Pattern Set

5.2.1 Input Generation. To evaluate the performance of our chip, we first choose the two-class classification problem of high-dimensional random binary patterns reported in Hussain et al. (2014), Poirazi and Mel (2001), and Hussain, Liu, and Basu (2015). A 40-dimensional random gaussian datum is mapped to a neurally plausible sparse $d = 400$-dimensional binary vector using 10 nonoverlapping receptive fields per dimension (Hussain et al., 2014). The widths of the receptive fields at each location were chosen such that they have equal probability of being active. Since 10 receptive fields span each dimension, each of them had a probability of 0.1 of being active in any given pattern. This sparse, high-dimensional vector is next converted to a spiking input by mapping the two binary values to Poisson spike trains

with mean firing rates of f_{HIGH} and f_{LOW}, respectively. This type of random binary spike pattern is also used to test the classification performance of other hardware neuromorphic circuits (Mitra, Fusi, & Indiveri, 2009). A $T_{pat} = 500$ ms sequence of pulse trains along with the input line addresses is generated for each of the $P = 200$ random patterns that are split equally and arbitrarily into the two classes. The network is first trained on the binary vectors (Hussain et al., 2015) to find the desired connection matrix on a PC, which is then downloaded to the FPGA for hardware evaluation. Next, the spike patterns are sent as input to the chip, and classification results are obtained for various configurations. A software implementation of the NNLD classifier is also used to evaluate performance on spike train inputs; we use that as a baseline to compare the performance of the hardware. In the following text, we use the term *simulation* to refer to this software implementation of NNLD classifier.

5.2.2 Classification Performance. In the first experiment, the values of f_{HIGH} and k were varied while keeping $f_{LOW} = 0$, margin $\delta = 0$, and dendritic leak $f_{leak} \approx 0$. The results for this are plotted in Figure 11 in terms of

$$\text{Classification error} = \frac{\text{Number of misclassified patterns}}{\text{Total number of patterns}} \tag{5.4}$$

expressed as a percentage. In general, we expect higher values of f_{HIGH} to be better since this increases the difference in current of an ON and an OFF synapse. This should make the system more robust to noise and mismatch. This trend was indeed found to be true for $k = 2, 3, 4, 5$, and 7 and correlates well with software simulations of the NNLD classifier as shown in Figure 11. However, for larger values of k, the increased current in the square law block and DPI synapse pushes the transistors to above threshold, making it deviate from desired translinear behavior. The second trend is observable by looking at the change in error with number of synapses per branch k (note that the total number of synapses is $2mk$). From simulations, Hussain et al. (2013) observed that increasing k initially reduces error due to more synaptic resources. But for very large k, testing error would increase since the large input current saturates the dendritic nonlinearity. The same trend is observed for the case of $f_{HIGH} = 62.5$ Hz, where the error reduces from $k = 2$ to $k = 10$ before increasing later. However, the increase in error at larger k is for a different reason here: because of the non–square law behavior of the dendritic branches at large currents. For larger values of f_{HIGH}, the increase in error starts occurring at lower values of $k \approx 7$ (a larger f_{HIGH} results in more current per synapse and hence fewer synapses are needed to push the square law block and synapse above threshold).

We mention that the number of unclassified patterns (those causing equal neuronal firing on both current difference inputs) is too insignificant

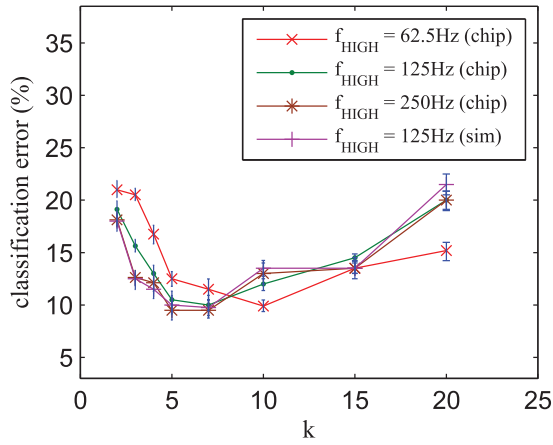
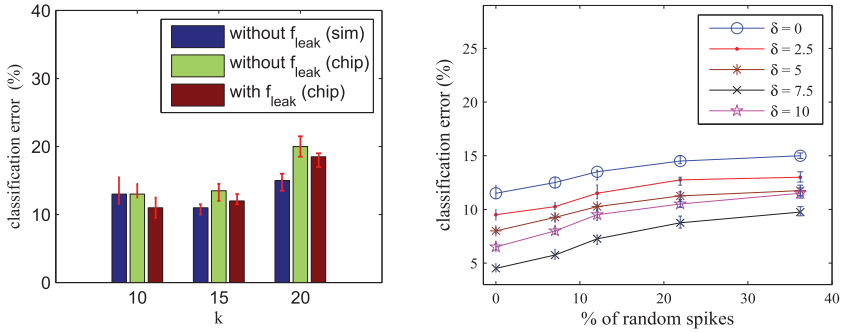


Figure 11: Measured classification error for different values of k and f_{HIGH}, taken over five trials. For each value of f_{HIGH}, increasing k initially reduces the error due to the higher computational power of more synapses. But eventually the error increases because the square nonlinearity block enters above threshold regime. Values of k between 7 and 10 are found to be optimal.

to be reported separately from Matlab simulations and is almost always zero in measurements.

To reduce this error, we next introduced dendritic leak currents that have been shown to be useful in reducing average currents (Hussain et al., 2015) to the dendrite and increase its effective dynamic range. The value of f_{leak} was set equal to $0.8 \times k \times f_{HIGH} \times p$ to cancel most of the common-mode current. Here, $p = 0.1$ denotes the probability of a randomly selected input dimension being high for a given pattern; it depends on the number of receptive fields used to generate the high-dimensional mapping. The desired value of V_{leak} for this f_{leak} is obtained from the curve fit shown in Figure 8b. It can be seen from Figure 12a that the classification error indeed reduces for $k = 10$, 15, and 20 by adding dendritic leak for $f_{HIGH} = 250$ Hz. However, the reduction is not as much as expected from software simulations due to the mismatch between the transistors M_{s7} in Figure 3b to create the leak currents. In the future, we can use a current splitter-based configurable leak current in each dendrite to remove the mismatch. Since this is a static setting (unlike if a splitter is used in a synapse), the dynamic performance of the splitter for small currents is not a bottleneck. Another option is to use the characterization results of this mismatch in the training process to find a new connection matrix. Finally, as we show next, we can partially reduce the effect of this mismatch by using margin-based training.

The effect of increasing margin δ during training was tested when noisy background spikes are added by choosing $f_{LOW} \neq 0$. The network is



(a) (b)

Figure 12: (a) The effect of increasing dendritic leak is to reduce the measured classification error for $f_{HIGH} = 250$ Hz by reducing the current entering the square block and preventing it from going into above-threshold regime. The data are averaged over five trials. (b) Increasing margin δ improves the robustness of the system to added random spikes for $k = 10$ and $f_{HIGH} = 125$ Hz observed over five trials. Increasing δ initially reduces the error to $\delta = 7.5$. Increasing δ beyond that increases error due to difficulty in training.

retrained for large margins with $k = 10$ using the adaptive δ algorithm in Hussain et al. (2015), and the new connection matrix is used for testing. The errors for the cases of $\delta = 0$, 2.5, 5, 7.5, and 10 are compared with an increasing number of random spikes in Figure 12b when $f_{HIGH} = 125$ Hz. It can be seen that for all noise levels (obtained by increasing f_{LOW}), the increased margin setting of $\delta = 7.5$ is the optimal setting in our case, achieving 6% to 7% less error than the nominal case of $\delta = 0$, proving that the added margin indeed helps in improving robustness. However, it can also be observed that the classification performance degraded slightly with an increase in the margin from $\delta = 7.5$ to $\delta = 10$ in both software simulation of NNLD and chip measurements. This is attributed to the fact that increasing margin beyond a point makes it difficult for the training process to converge. We can also use the measured result of 4.5% error (for $\delta = 7.5$ and no random spikes) to benchmark the performance of the chip with other classifiers.

A software implementation of the NNLD achieves a comparable 4.5% error for the same parameter setting. To compare with other non-spike-based classifiers in software, we modified the binary inputs by adding noise to approximate the situation of noisy spike trains. The variance of noise was set so that the performance of NNLD on these binary inputs matches its software performance on spike train inputs. With these noisy binary patterns, a perceptron classifier achieves 9% error, while an extreme learning Machine

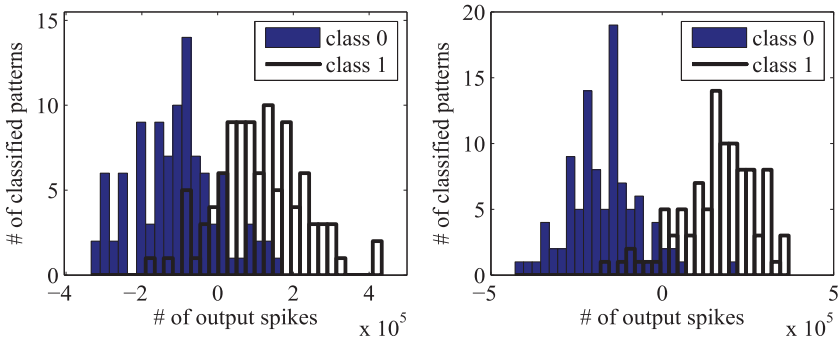


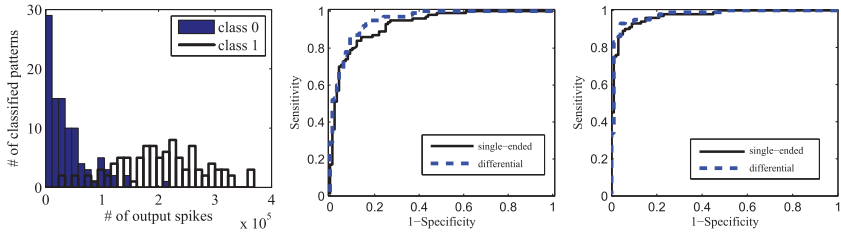
Figure 13: Measured distribution of the difference of output spike counts showing that increasing margin δ from (a) 0 to (b) 7.5 increases the separation between the spike count distributions of the two classes. In this case, $k = 10$ and $f_{HIGH} = 125$ Hz.

(ELM; Huang, Zhu, & Siew, 2006) achieves 4% error with 2000 hidden neurons. Note that the NNLD uses only 160 binary synapses, while the perceptron uses 400 high-resolution weights. The ELM uses 2000 high-resolution weights and close to 1 million random weights. This shows the benefit of our approach over networks with weighted synapses.

Finally, note that the value of δ used here follows the notation of Hussain et al. (2015) introduced in section 2, where the weight of a synapse is normalized to 1 and $K_{neu} = 1$. To translate this value to a margin δ_I in terms of input current to the I&F neuron, we can apply the following transformation,

$$\delta_I = \frac{\overline{I_{syn}}^2}{I_{STHR}} \times x_{thr} \times \delta, \tag{5.5}$$

where $x_{thr} = 2$ in the software model used for training. This can also be converted to an equivalent margin in terms of frequency at the output of the neuron by multiplying with K_{neu} for the IC. This points to another way to observe the effect of increased margin: plot the distribution of output spike count differences that is proportional to $f(I_P^z - I_N^z) - f(I_N^z - I_P^z)$ in equation 2.6, without taking the IIN steady-state current into account. Applying the Heaviside function $g()$ on this quantity gives the classification output. This is shown in Figures 13a and 13b for $k = 10$, $f_{HIGH} = 125$ Hz, and two values of $\delta = 0$ and 7.5, respectively. The x-axis shows the difference in number of spike counts of the two cells, which is equal to $f(I_P^z - I_N^z) - f(I_N^z - I_P^z)$. It can be seen that increasing the value of δ also increases the separation between



(a) (b) (c)

Figure 14: (a) Measured distribution of output spikes proportional to $f(I_P^z - I_N^z)$ in equation 5.6 for the case of $k = 10$, $f_{HIGH} = 125$ Hz, and $\delta = 7.5$. (b) ROC curve for $\delta = 0$. Other parameters remain same. Here, AUC averaged over four trials for two outputs is 0.95 compared to the earlier result of 0.93 for a single output, showing more benefits of considering balanced differential outputs in low-margin cases. (c) ROC curves for the case of $\delta = 7.5$, $k = 10$, $f_{HIGH} = 125$ Hz showing that the differential architecture with two outputs performs similarly (AUC = 0.978 averaged over four trials) compared to classifying based on a single output $f(I_P^z - I_N^z)$ only (AUC = 0.969 averaged over four trials).

the distributions for class 0 (blue solid bars) and class 1 (bold white bars). This is the reason for better classification performance with increased δ.

An interesting option is to classify the patterns based on only one of the differential outputs $f(I_P^z - I_N^z)$ in equation 2.6, again without taking the IIN steady-state current into account. In that case, a threshold thr has to be set on the output spike counts, and we declare a pattern as belonging to class 1 if $f(I_P^z - I_N^z) > thr$. Thus, effectively the classification equation becomes

$$y = g[f(I_P^z - I_N^z) - thr]. \tag{5.6}$$

This would still have the advantage of rejecting common-mode currents but would not need two output neurons. However, the value of thr will need to be changed if system parameters like m, k and f_{HIGH} change. We will refer to this type of architecture as single-output differential (the original differential architecture in Figure 2a is henceforth referred to as double output). We plot the distribution of output spikes for this type of architecture in Figure 14a and set $k = 10$, $f_{HIGH} = 125$ Hz, and $\delta = 7.5$ to compare with Figure 13b. It can be seen from the distribution that a large number of patterns belonging to class 0 generate zero output spikes. Next, we vary the threshold for both the single-output and double-output differential cases and plot the receiver operating characteristic (ROC), a commonly used metric to compare classifiers (Mitra et al., 2009). We plot the ROC for two different values of δ—0 and 7.5—in Figures 14b and 14c, respectively. It can be seen that the ROC for the differential case is slightly higher than the single-ended case

in Figure 14b, indicating better performance for the former. To quantify this difference, we use area under the curve (AUC) as the metric and find that the double-output differential one has an AUC of 0.95 compared to 0.928 for a single-output differential. However, for the higher-margin case in Figure 14c, the similar AUC values of 0.978 and 0.969 for double and single outputs, respectively, show that margin enhancement makes both cases similar. This indicates that if a small performance penalty is acceptable and the threshold can be manually set, single-output differential architectures should be used since they eliminate the need for one I&F neuron. This is one additional benefit of margin enhanced learning.

5.3 Pattern Classification: UCI Data Sets

5.3.1 Data Set and Input Generation. To evaluate the classification performance of our system on real-world data sets, we next tested it on two standard UCI data sets: the Breast Cancer (BC) data set and the Heart data set. These were also two-class classification problems; however, they differ from the former experiments in that the random data sets tested only the noise resilience property of the NNLD since the training patterns were converted to noisy spike trains for testing. In the UCI data sets, training and testing use different patterns and, hence, evaluate the generalization capability of our chip. For both data sets, each input vector was mapped to a higher-dimensional sparse binary vector. Similar to the method in section 5.2.1, this mapping was done by employing 10 nonoverlapping receptive fields to span each of the original dimensions of the data. The width of each receptive field was again chosen so that all of the higher dimensions have an equal probability of being active. Hence, the original 9 and 13 dimensions of the BC and Heart data sets were mapped to 90 and 130, respectively. The numbers of training and testing samples for the BC data set are 222 and 383, respectively, while the corresponding numbers for the Heart data set are 70 and 200, respectively.

5.3.2 Data Set and Input Generation. Table 2 compares the performance of different classifiers on these data sets. For the NNLD classifier, k was fixed at 7, and a margin of $\delta = 7.5$ was used, with $f_{HIGH} = 250$ Hz, because these settings yielded the best results for the random binary pattern classification case. Five trials were conducted on each UCI data set, where the NNLD was separately trained in software with different initial conditions. Also, different instances of Poisson spike trains were generated in each trial. For the NNLD case, we report the result of software simulation on binary inputs and spike inputs to show the loss in performance expected due to mapping to noisy spike trains. The results for SVM and ELM are taken from Babu and Suresh (2013). We also show the results for software implementation of a perceptron to classify the same high-dimensional binary patterns that are input to the NNLD. We find that for all cases, software implementation

Table 2: Classification Performance on UCI Data Sets.

	SVM			ELM			Perceptron			NNLD				
Data Sets	N	W	A(%)	N	W	A(%)	N	W	A(%)	D	W	A(%) (binary, software)	A(%) (spike, software)	A(%) (chip)
BC	24	240	96.7	66	660	96.4	1	90	87.7 ± 0.97	16	112	96.7 ± 0.97	96.1 ± 0.14	94.7 ± 1.05
Heart	42	588	75.5	36	504	76.5	1	130	76.5 ± 0.98	16	112	79.3 ± 1.05	77.0 ± 2.44	76.6 ± 3.61

Note: N: number of neurons; W: number of weights; D: number of dendrites; A(%): classification accuracy in percentage.

of the NNLD achieves performance comparable to SVM or ELM and superior to that of the perceptron. Clearly, the network we have proposed performs well compared to the more extensive networks constructed with weighted synapses (high-resolution weights). In all cases, there is a drop of approximately 1% in accuracy in software when the binary inputs are mapped to noisy spike trains. Finally, the measured accuracy of the IC is approximately 1% to 2% less than the spike testing accuracy in software. It should be noted that the number of weights our NNLD used are two to six times fewer compared to SVM or ELM and comparable to that needed by the perceptron. But the perceptron used high-resolution weights, while the NNLD uses only binary weights, underlining its higher computational power.

The classification performance of the IC on the Heart data set is similar to performance of the perceptron in software. This is due to limitations on maximum m and k values we could set in this IC. However, this can be partially overcome by the concept of boosting (Mitra et al., 2009), where the output of several such NNLD classifiers may be combined to produce the final decision by voting. If each NNLD makes mistakes on different subsets of the pattern, the combined vote should result in far fewer errors. Of course, this comes at the expense of more dendrites and synapses. We tried this on the Heart data set by summing the spike counts of the P-cells and the N-cells separately for the five trials before comparing them. The resulting accuracy was 80%—better than those obtained from a single run of classification as well as those from other classifiers. But the effective number of synapses used increases to 560, which is similar to SVM and ELM. However, the synapses for NNLD are binary compared to high-resolution tunable weights required in other approaches.

5.4 Pattern Classification: Classification Speed. Finally, in order to gauge the speed of classification (trade-off with accuracy) achievable by the classifier, it was tested again on the standard BC data set. This was approached by shrinking the time window of observation (T_{pat}) of an individual pattern. The advantage of using a smaller time window for classification is greater classification speed, in addition to lower firing of the output neuron and, hence, lower energy dissipated. Two cases were considered.

5.4.1 Poisson Spike Train. Decreasing the time window indiscriminately has the demerit of introducing noise in the input, and this trade-off was tested, as we describe next.

Data set and input generation. The input data set generation followed the same principle as described in section 5.3.1: each input vector was mapped to a higher-dimensional sparse binary vector, and each resulting vector converted to a Poisson spike train, with the binary value 1 mapped to a train of mean firing rate f_{HIGH} and 0 to a firing rate $f_{LOW} = 0$. The training and the testing sets differ. The learning or training of the classifier followed.

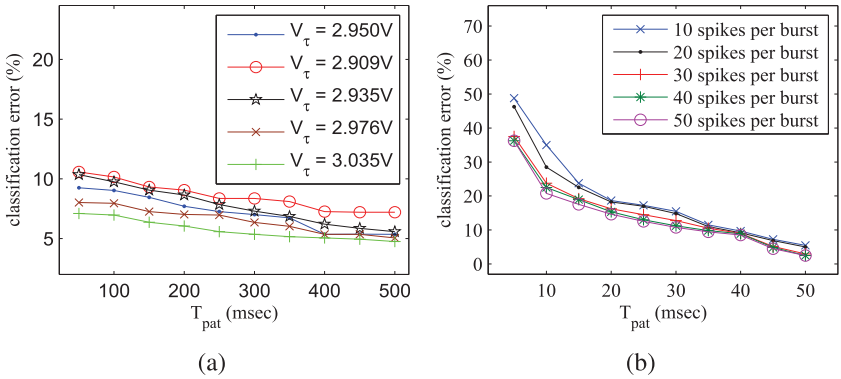


Figure 15: (a) Effect of gradually increasing the classification time interval or window is to reduce the measured error for $f_{HIGH} = 250$ Hz by reducing the noisy spikes in the output spike train through longer integration. Increasing bias V_τ also results in lower error due to more filtering, as well as more synaptic current for an active afferent raising the difference between an ON and an OFF synapse. (b) Classification accuracy generally improves with an increasing time window, a trend observed in panel a, as well as with greater spike density per input burst per active afferent, again due to a more detectable difference between the currents of an ON and an OFF synapse.

Classification performance. The pulse train sequence fed to the network had different values of T_{pat}, varying from $T_{pat} = 50$ ms to $T_{pat} = 500$ ms in steps of 50 ms. In each case, 383 binary patterns in the testing subset of the BC data set were classified with $k = 7$ and $f_{HIGH} = 250$ Hz, these being the optimal settings from the earlier runs. In addition, for each case of T_{pat}, the bias V_τ was also swept over a range. The expected trend is found to be true from Figure 15a, which shows error to gradually decrease and finally become almost constant with increasing classification time intervals; the errors also become lower with increasing V_τ for a particular T_{pat}. For example, at the highest setting of V_τ, the optimal performance corresponding to a 9% classification error is reached within 300 ms.

The trend with V_τ-variation is easily explained: with increasing bias, the synaptic time constant τ_s (see section 3.1) increases, resulting in more averaging of the noisy Poisson input. Also, current from a synapse on an active afferent rises. This results in a greater difference between an ON and an OFF synapse. These two effects combine to produce lower error at higher setting of V_τ.

5.4.2 Single Spike Burst.

Data set and input generation. The input generation for this two-class classification problem differed from that in section 5.2.I. After the mapping with

the nonoverlapping receptive fields to expand each of the original dimensions of the data, the resulting high-dimensional vector is converted to a spiking input by mapping the binary value 1 to a burst of spikes concentrated over a very small time duration (2 msec), and the value 0 to no spikes at all.

Classification performance. The classification was conducted through a series of runs, each of which corresponded to a different spike count in the input burst per active afferent of the classifier. Each run itself measured the performance over increasing classification time windows (or pattern duration) T_{pat}, which was kept at a maximum of 50 ms this time. As is evident, this is substantially lowered from the previous value of 500 ms, because the sole purpose of this experiment was to see how quickly our system could classify a pattern set with reasonable accuracy (hence, only a small spike burst instead of an entire noisy spike train applied to a binary 1-afferent). Figure 15b shows the experimental results. Again, we find improved performance with expanding classification time in each case, similar to the trend obtained in section 5.4.1. The drastic reduction in accuracy at values of $T_{pat} < 10$ ms is because of the finite number of spikes obtainable in such a short time. This limits the discriminative ability of the neuron. It might be noted that the general classification accuracy improves with increasing spike density per burst as well; this is intuitively agreeable because more spikes per input burst per active afferent implies a greater difference in the current of an ON and an OFF synapse (as discussed in section 5.2.2). In this case, we reach an error of 5% at $T_{pat} = 50$ ms with 20 spikes per burst.

6 Discussion

6.1 Relation to Other Work. In terms of spike-based classifiers in hardware, our results of 4.5% error in classifying 200 random binary patterns in hardware using only 160 binary synapses show much better performance compared to other systems. For example, the work in Mitra et al. (2009) could classify about 12 random binary patterns at an error of 20% using 1200 binary synapses; of course, they included learning capabilities on-chip that are not present in our work. In terms of VLSI implementations of dendritic function, Nease, George, Hasler, Koziol, and Brink (2012) present a voltage mode diffuser circuit to model passive properties of dendrites. They show the ability of this circuit to exhibit properties matching cable theory. Wang and Liu (2010, 2013) present a current mode implementation of active dendrite models and use it to study its response to spatiotemporal spike inputs. Compared to these detailed models, our model of dendritic nonlinearity is much more simplified. Instead, we have focused on using this model for an application of spike-based pattern classification. Our results show the viability of using binary synapses for pattern classification when coupled with margin-based learning to counter the imperfections of VLSI

implementations. The key to achieving good performance is to have a larger ratio of nonlinear to linear processing—short linear summation of synaptic inputs have to be passed through a nonlinearity before further summations.

This method can be used by other analog neuromorphic processors (Brink et al., 2013; Qiao et al., 2015) where the dendritic nonlinearity can be replaced by neurons, while the software model used for training uses structural plasticity to choose the few best connections per neuron, which can then be incorporated into look-up tables for AER. The inherent learning capability in some of these systems can be used to fine-tune the performance further. Here, we assume that the benefits available in the NNLD network are due to the additional nonlinearities provided by the dendrites. Hence, it should be possible to get similar benefits by using neuronal nonlinearities in place of dendrites. However, in that case, the area would increase a lot since the dendrite circuit does not have capacitors and is more compact than an I&F neuron, which has at least one capacitor for membrane dynamics and more for refractory period and spike frequency adaptation implementations. Finally, recently Spiess, George, Cook, and Diehl (2016) have explored the use of structural plasticity to denoise neuronal responses. However, no real-world classification problems were reported, and no hardware measurements were done. We believe our work is the first to show such results.

6.2 Future Work. Though the initial results from this simple chip are promising, it has certain limitations that can be improved in future designs. We discuss these aspects and other extensions in this section.

In this proof-of-concept work, since the WTA for comparison was not implemented, the number of spikes from the output neuron was not regulated down. On average, the neurons fired about 15,000 spikes in a pattern duration of 500 ms. However, not all of these spikes are necessary, and far fewer spikes can give comparable accuracy if an inhibitory interneuron (IIN)-based WTA is included on the chip. This has already been depicted in Figure 2a for a differential architecture. From this representation, it is clear that the IIN steady-state current is fed back on a negative feedback path to the inputs of both neurons, so that for class 1 patterns, current $(I_N^z - I_P^z)$ would be rapidly decreased to zero, with a simultaneous decrease of the current $(I_P^z - I_N^z)$, while for class 0 patterns, the reverse would occur. This would suggest a significant (possibly greater than 50%) reduction in spiking of the output neuron. The mathematical formulation in equation 2.6 also supports this statement if one notes that the current $(I_N^z - I_P^z)$ is not the negative of the current $(I_P^z - I_N^z)$; rather, it is the representation of the difference between the N-cell current and the P-cell current with the connectivities swapped. Obviously, for a pattern to be classified in class 1, $(I_P^z - I_N^z)$ must exceed $(I_N^z - I_P^z)$, and vice versa for classification in class 0.

To evaluate the possible benefits, we implemented a software model of the WTA and varied the parameters τ_{IIN} and $I_{0,IIN}$ denoting the time

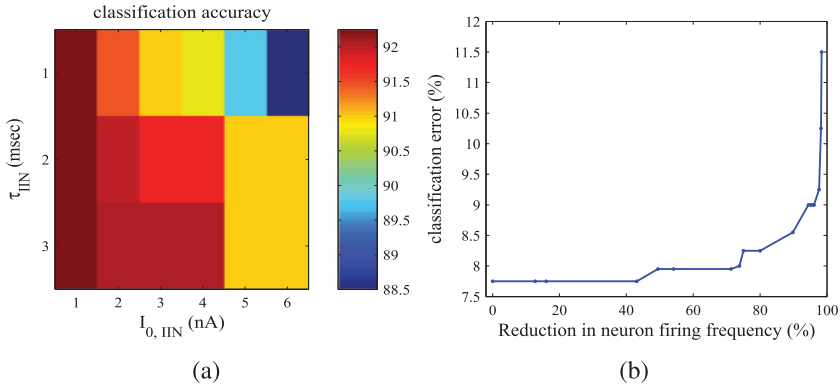


Figure 16: (a) Change in classification accuracy when the time constant and peak amplitude of the inhibitory postsynaptic current in the WTA are varied. (b) The data in the earlier simulation are replotted to show change in accuracy against the percentage reduction of spikes. Compared to the original accuracy of 7.75%, slightly lower accuracies of 7.95% and 9% can be obtained with a concomitant reduction of 73% and 96%, respectively, in the number of spikes.

constant and peak amplitude of the inhibitory postsynaptic current generated by the interneuron. The performance of the NNLD classifier with $k = 10$, $m = 8$, and $\delta = 0$ was evaluated for the case of random binary inputs mapped to spike trains with $f_{HIGH} = 250$ Hz. As shown in Figure 16a, different combinations of τ_{IIN} and $I_{0,IIN}$ lead to different error rates; errors increase with higher $I_{0,IIN}$ as an increasing proportion of the output neuronal spikes gets eliminated from the negative feedback. This data are replotted in Figure 16b to show the trade-off in terms of accuracy versus spike rate. It can be seen that compared to the original accuracy of 7.75%, slightly lower accuracies of 7.95% and 9% can be obtained with a concomitant reduction of 73% and 96%, respectively, in the number of spikes. This points to the great benefit of having an appropriately tuned inhibitory interneuron and a way to lower output neuronal firing without cutting down on the classification interval; it will be included on chip in future versions of the system.

The current chip has a limited number of dendrites ($m = 8$) per cell. Combined with the constraint on the maximum number of synapses per dendrite (posed by transistors in the DPI synapse and squaring block going out of subthreshold regime as shown in equations 3.5 and 3.3), this leads to limited computational capability of each cell. In future versions of the chip, we will increase the size of the output transistor of the DPI synapse, as well as the transistors in the translinear loop of the square block to get a higher operating range. Architecturally, we will move to cells with more dendrites since that also increases the computational power of the classifier (Hussain et al., 2015). Future versions of the system will integrate multiple

NNLD per chip with a fully asynchronous AER-based input and output interface so that multiple chips can be tiled to create bigger processors. In this kind of system, we can also use the concept of boosting (shown in section 5.3.2 to improve performance) to allocate multiple NNLD to decide on a class by voting. Mismatch between different branches reduces the accuracy of the hardware (though margin-based learning is useful to counter this effect). We will use the results of chip characterization to modify the software model during training. The new connection matrix, where the learning accounts for the mismatch, should produce a better match in results between hardware and software. We also plan to extend our hardware to include on-chip learning of connection matrices that can account for all mismatches in the hardware directly. Initial architectural explorations in this direction have already been done (Roy, Kar, & Basu, 2014) and a chip is being fabricated to test this idea.

In terms of algorithms, it has already been shown that the NNLD-based classifier can easily be extended to multiclass problems (Hussain et al., 2014) as well as to spike-time-based pattern classification (Roy, Banerjee et al., 2014; San, Basu, & Hussain, 2014). We will employ future systems to perform classification of handwritten digits from the MNIST database, a commonly used multiclass classification standard in current neuromorphic spike-based algorithms (O'Connor et al., 2013; Neftci, Das, Pedroni, Kreutz-Delgado, & Cauwenberghs, 2013). This problem is still considered a mean-rate-encoded system. To assess the ability of the hardware to identify spike-time-based patterns, we will employ it to classify the spike trains from a liquid state machine as done in Roy, Banerjee et al. (2014) and random spike latency patterns as done in Gutig & Sompolinsky (2006). We also plan to connect these ICs to real-time spike-generating auditory sensors (Chan et al., 2007) to perform rapid speech classification in low power.

7 Conclusion

In this letter, we have presented the VLSI circuit design in 0.35 μm CMOS of a neuromorphic spike-based classifier with eight nonlinear dendrites per neuron and two opponent neurons per class. We presented characterization results to prove the functionality of all subblocks from a V_{dd} as low as 1.8 V and also some results of classifying complex spike-based high-dimensional binary patterns. The classification error in classifying 200 patterns randomly assigned to two classes was obtained under different conditions. We showed that the addition of a dendritic leak and classification margin helps improve performance and makes the system robust against noise. With an optimal classification margin, the hardware system performs comparably to SVM and ELM on two UCI data sets and with far fewer binary weights. We also demonstrated that the margin enhancement algorithm allows single-output differential operation with similar accuracy as double output, thus allowing us to reduce the number of neurons.

Pattern classifications within 50 ms were possible while using bursts of spikes to represent input binary value. The accuracy was degraded compared to the software due to transistor mismatch and non–square law behavior of the dendritic nonlinearity at high currents. Future work will use calibration to improve accuracy and include WTA on chip to reduce the output spike counts. We will also employ future generations of this chip for multiclass and spike time based classifications.

Appendix: Details of the FPGA Controller

The FPGA controller consists of the following blocks:

A.1 Block Memory (Module A). Module A, shown in Figure 6, is the volatile memory array implemented in the hardware. For generating presynaptic pulses, the input spike train vector (*spkTm*), input line addresses (*spkAd*), and connectivity matrix created in the pattern classification program are transferred from a PC to block memory A, block memory B, and block memory C, respectively. The structures of the *spkTm* and *spkAd* vectors are shown in Figure 17. Since *spkTm* is a sparse array of binary firing events, only those time instances of *spkTm* having binary value 1 are stored in block memory A. These time instances, T_is, are generated with respect to an FPGA internal clock frequency (f_A). Similarly, input addresses AD_i at these particular time instances are stored in block memory B. These two parts store input information and will not be needed while interfacing with a real spiking sensor.

The connectivity matrix of this neural network has the dimension of $m \times d$ (as discussed in section 2), where d is the number of input lines and m is the number of dendrites. Since the number of connections per dendrite $k \ll d$, this matrix is also sparse in nature. Storing the entire content of the sparse matrix as a look-up table (LUT) would require memory space on the order of $m \times d$ and is wasteful. Instead, we store only the addresses of nonzero $m \times k$ elements as a linear array *synCol*, as shown in Figure 18 where $R = m \times k$ and s_j denote decoder addresses. For example, if the first input line connects to dendrites 2, 3, and 5, the first three entries of *synCol* will be $s_1 = 2$, $s_2 = 3$, and $s_3 = 5$. But now, given an input pulse address, determining which decoder addresses have to be generated is a bit difficult since the input pulse address cannot be used to directly index into this array. To circumvent this, we store another linear integer array, *addrPtr*, with d entries. The ith entry in *addrPtr* stores the address a_i of the *synCol* array where the first dendritic connection (decoder address) for the ith line is stored. The number of dendritic connections n_i for input line i is obtained as $a_{i+1} - a_i$. The controller produces a pulse on the line *SPK* for each of the n_i decoder addresses stored at entries a_i to $a_{i+1} - 1$ of *synCol*. Now the memory space requirement is reduced to $m \times k + d$, which is much smaller than $m \times d$.

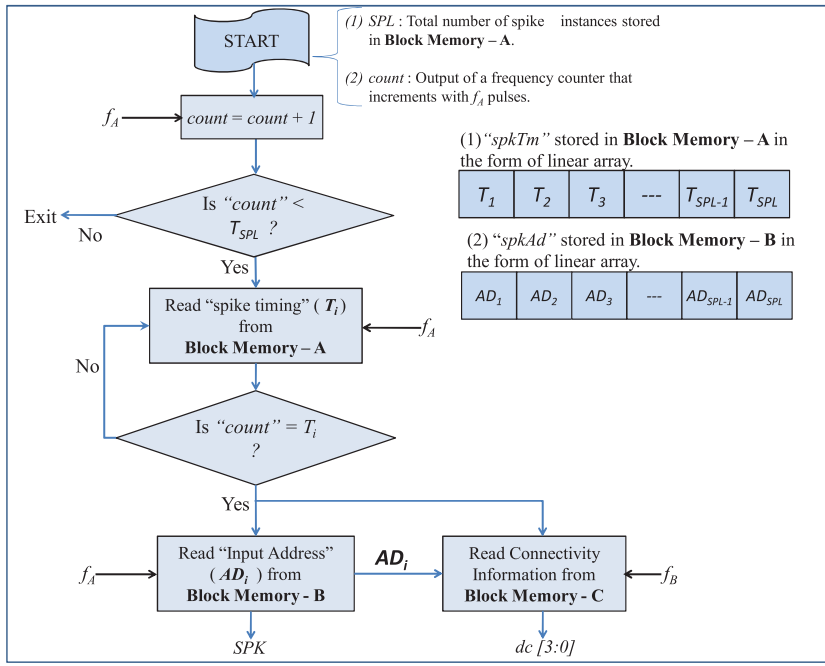


Figure 17: Flowchart depicting the pulse and address generation algorithm implemented on FPGA.

A.2 Clock References (Module B). Module B is the central clock generation and distribution block that generates reference clocks to other different functional modules. A software-controlled *START* command activates module B, which starts generating output clocks f_A, f_B, and f_C using an on-board PLL ($f_{pll} = 66.7$ MHz).

A.3 Synchronous Controller (Module C). Figure 17 shows the control logic implementation of module C to generate controlled spike trains to excite the neuromorphic IC. After the *START* command is executed, module C starts reading spike timing and the corresponding address from block memory A and block memory B. The sampling clock (f_A) is used for the read access of the memory. Module C reads the input address of an incoming pulse and generates 4 bit decoder values $dc[3:0]$ based on the connectivity matrix stored in block memory C. The logic is shown in Figure 18. Decoder bits $dc[3:0]$ at the FPGA output selects the correct dendritic branch to excite the corresponding synapse. While generating pulses, the controller also takes care of the weight of each synapse. For synaptic connections having integer weight N ($N > 1$), the controller generates N synaptic pulses to the IC.

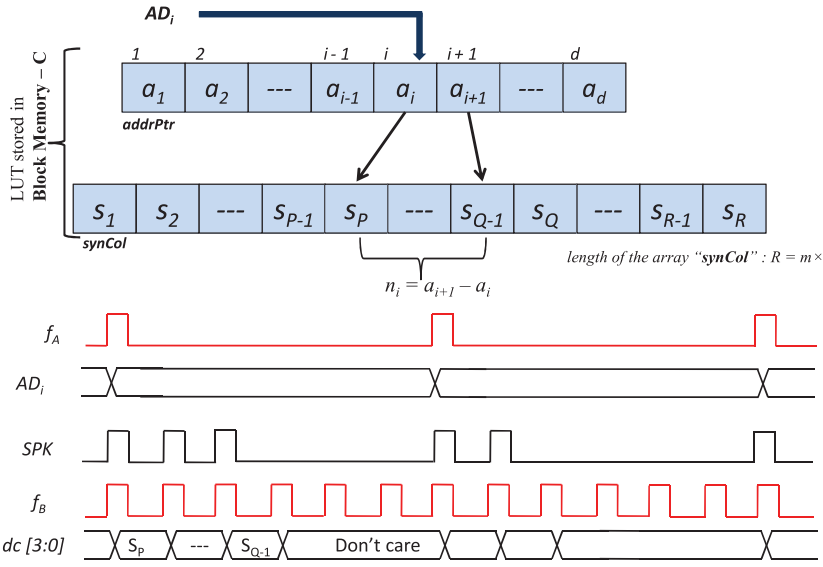


Figure 18: On receiving input address AD_i, a two-level look-up table (LUT) generates the decoder addresses (s_P to s_{Q-1}) of corresponding dendritic branches. These events are then sent to the IC on the *SPK* bus using a fast clock f_B.

A.4 REQ-ACK Controller (Module D). The block module D in Figure 6 is a shift register implemented for handshaking with the IC. The shift register is also driven by the high-frequency clock f_C. This module waits for the *REQ* signal generated by the IC to become high and deasserts the *ACK* signal.

A.5 REQ Pulse Counter (Module E). This block is a digital counter clocked by *REQ* pulses from the neuron and is indicative of output pulse frequency. This count information is later transferred to the PC to determine the result of classification.

Acknowledgments

We acknowledge financial support from MOE through AcRF Tier 2 grant ARC 8/13.

References

Arthur, J., & Boahen, K. (2007). Synchrony in silicon: The gamma rhythm. *IEEE Transactions on Neural Networks*, *18*(6), 1815–1825.

Babu, G., & Suresh, S. (2013). Sequential projection-based metacognitive learning in a radial basis function network for classification problems. *IEEE Transactions on Neural Networks*, *24*(2), 194–206.

Banerjee, A., Kar, S., Roy, S., Bhaduri, A., & Basu, A. (2015). A current-mode spiking neural classifier with lumped dendritic nonlinearity. In *Proceedings of the IEEE Intl. Symp. on Circuits and Systems*. Piscataway, NJ: IEEE.

Bartolozzi, C., & Indiveri, G. (2007). Synaptic dynamics in analog VLSI. *Neural Computation*, *19*(10), 2581–2603.

Be, J. V. L., & Markram, W. (2006). Spontaneous and evoked synaptic rewiring in the neonatal neocortex. *Proceedings of the National Academy of Sciences of the United States of America*, *103*(35), 13214–13219.

Boahen, K. (2000). Point to point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems-II*, *47*(5), 416–434.

Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., & Degnan, B. (2013). A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Transactions on Biomedical Circuits and Systems*, *7*(1), 71–81.

Chan, V., Liu, S., & Schaik, A. (2007). AER EAR: A matched silicon cochlea pair with address event representation interface. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(1), 48–59.

Culurciello, E., Etienne-Cummings, R., & Boahen, K. (2003). A biomorphic digital image sensor. *IEEE Journal of Solid-State Circuits*, *38*(2), 281–294.

Douglas, R., Martin, K., & Whitteridge, D. (1989). A canonical microcircuit for neocortex. *Neural Computation*, *1*(4), 480–488.

Gutig, S., & Sompolinsky, W. (2006). The tempotron: A neuron that learns spike timing–based decisions. *Nature Neuroscience*, *9*(1), 420–428.

Hasler, J., & Marr, B. (2013). Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in Neuroscience*, *7*(118).

Hsieh, H. Y., & Tang, K. T. (2012). VLSI implementation of a bio-inspired olfactory spiking neural network. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(7), 1065–1073.

Huang, G.-B., Zhu, Q.-Y., & Siew, C. K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, *70* 489–501.

Hussain, S., Basu, A., & Liu, S. C. (2014). Improved margin multi-class classification using dendritic neurons with morphological learning. In *Proceedings of the IEEE International Symposium on Circuits and Systems* (pp. 2640–2643). Piscataway, NJ: IEEE.

Hussain, S., Gopalakrishnan, R., Basu, A., & Liu, S. C. (2013). Morphological learning: Increased memory capacity of neuromorphic systems with binary synapses exploiting AER based reconfiguration. In *Proceedings of the IEEE Intl. Joint Conference on Neural Networks*. Piscataway, NJ: IEEE.

Hussain, S., Liu, S. C., & Basu, A. (2015). Biologically plausible, hardware-friendly structural learning for spike-based pattern classification using a simple model of active dendrites. *Neural Computation*, *27*(4), 845–897.

Indiveri, G. (2003). A low-power adaptive integrate-and-fire neuron circuit. In *Proceedings of the IEEE Intl. Symp. on Circuits and Systems* (vol. 4, pp. 820–823). Piscataway, NJ: IEEE.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128 × 128 120 dB 15 us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, *43*(2), 566–576.

Maass, W. (1999). *Pulsed neural networks*. Cambridge, MA: MIT Press.

Maass, W., & Markram, H. (2004). On the computational power of circuits of spiking neurons. *Journal of Computer and System Sciences*, *69*(4), 593–616.

Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, *78*(10), 1629–1636.

Mitra, S., Fusi, S., & Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Transactions on Biomedical Circuits and Systems*, *3*(1), 32–42.

Nease, S., George, S., Hasler, P., Koziol, S., & Brink, S. (2012). Modeling and implementation of voltage-mode CMOS dendrites on a reconfigurable analog platform. *IEEE Transactions on Biomedical Circuits and Systems*, *6*(1), 76–84.

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., & Cauwenberghs, G. (2013). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, *7*(272).

O'Connor, P., Neil, D., Liu., S. C., Delbruck, T., & Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, *7*, 1–13.

Oster, M., Douglas, R., & Liu, S. C. (2009). Computation with spikes in a winner-take-all network. *Neural Computation*, *21*(9), 2437–2465.

Poirazi, P., & Mel, B. W. (2001). Impact of active dendrites and structural plasticity on the memory capacity of neural tissue. *Neuron*, *29*(3), 779–796.

Polsky, A., Mel, B. W., & Schiller, J. (2004). Computational subunits in thin dendrites of pyramidal cells. *Nature Neuroscience*, *7*(6), 621–627.

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., & Indiveri, G. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Frontiers in Neuroscience*, *9*(141).

Rieke, F., Warland, D., de Ruyter van Steveninck, R., & Bialek, W. (1999). *Spikes: Exploring the neural code*. Cambridge, MA: MIT Press.

Roy, S., Banerjee, A., & Basu, A. (2014). Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations. *IEEE Transactions on Biomedical Circuits and Systems*, *8*(5), 681–695.

Roy, S., Basu, A., & Hussain, S. (2013). Hardware efficient, neuromorphic dendritically enhanced readout for liquid state machines. In *Proceedings of the IEEE Biomedical Circuits and Systems* (pp. 302–305). Piscataway, NJ: IEEE.

Roy, S., Kar, S., & Basu, A. (2014). Architectural exploration for on-chip, online learning in spiking neural networks. In *Proceedings of the 14th Intl. Symp. on Integrated Circuits* (pp. 128–131). Piscataway, NJ: IEEE.

San, P. P., Basu, A., & Hussain, S. (2014). Hardware-friendly morphological learning for spiking-neuron with dendritic nonlinearity. In *Proceedings of the IEEE Intl. Joint Conference on Neural Networks*. Piscataway, NJ: IEEE.

Spiess, R., George, R., Cook, M., & Diehl, P. U. (2016). Structural plasticity denoises responses and improves learning speed. *Frontiers in Neuroscience*, *10*(93).

Stettler, D. D., Yamahachi, H., Li, W., Denk, W., & Gilbert, C. D. (2006). Axons and synaptic boutons are highly dynamic in adult visual cortex. *Neuron*, *49*(6), 877–887.

Trachtenberg, J. T., Chen, B. E., Knott, G. W., Feng, G., Sanes, J. R. Welker, E., & Svoboda, K. (2002). Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature*, *420*, 788–794.

Wang, R., Hamilton, T. J., Tapson, J., & van Schaik, A. (2014). A mixed-signal implementation of a polychronous spiking neural network with delay adaptation. *Frontiers in Neuroscience*, *8*.

Wang, Y., & Liu, S. C. (2010). Multilayer processing of spatiotemporal spike patterns in a neuron with active dendrites. *Neural Computation*, *22*, 2086–2112.

Wang, Y., & Liu, S. C. (2013). Active processing of spatio-temporal input patterns in silicon dendrites. *IEEE Transactions on Biomedical Circuits and Systems*, *7*(3), 307–318.

Received February 21, 2017; accepted September 18, 2017.