DELTRON : neuromorphic architectures for delay based learning

Hussain, Shaista; Basu, Arindam; Wang, Mark; Hamilton, Tara Julia

2012

Hussain, S., Basu, A., Wang, M., & Hamilton, T. J. (2012). DELTRON : neuromorphic architectures for delay based learning. 2012 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp.304-307.

<https://hdl.handle.net/10356/101646>

<https://doi.org/10.1109/APCCAS.2012.6419032>

© 2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<http://dx.doi.org/10.1109/APCCAS.2012.6419032>].

Downloaded on 26 Apr 2023 09:35:35 SGT

DELTRON: Neuromorphic Architectures for Delay based Learning

Shaista Hussain, Arindam Basu

School of EEE, Nanyang Technological University,
Singapore

shaista@ntu.edu.sg, arindam.basu@ntu.edu.sg

Mark Wang^{*}, Tara Julia Hamilton⁺

⁺Univ. of New South Wales, ^{*}Univ. of Western Sydney
Australia

t.hamilton@unsw.edu.au

Abstract— We present a neuromorphic spiking neural network, the DELTRON, that can remember and store patterns by changing the delays of every connection as opposed to modifying the weights. The advantage of this architecture over traditional weight based ones is simpler hardware implementation without multipliers or digital-analog converters (DACs). The name is derived due to similarity in the learning rule with an earlier architecture called Tempotron. We present simulations of memory capacity of the DELTRON for different random spatio-temporal spike patterns and also present SPICE simulation results of the core circuits involved in a reconfigurable mixed signal implementation of this architecture.

I. INTRODUCTION

Neuromorphic systems emulate the behavior of biological nervous systems with the primary aims of providing insight into computations occurring in the brain as well as enabling artificial systems that can operate with human-like intelligence at power efficiencies close to biology. Though initial efforts were mostly limited to sensory systems [1,2], the focus of research has slowly shifted towards the implementation of functions of higher brain regions like recognition, attention, classification etc. However, most of the previous researchers have primarily focused on the implementations of somatic nonlinearity, compact learning synapses and address event representation (AER) for asynchronous communication [3-5]. As a result, there is a need for modeling and understanding the computational properties of other components of our nerves: the axons and dendrites which have been largely ignored till now.

This is also facilitated by recent experimental and computational work which has shed light on possible computational role of these structures. For example, Izhikevich showed the phenomenon of ‘polychronization’ [6] which uses a combination of axonal delays and spike time dependent plasticity (STDP) to remember a large number of spatio-temporal spike patterns. This has spurred a renewed interest in the possible role of conduction delays and has even led to analog VLSI implementations of delay models of axons [7].

In this paper, we also focus on a possible computational architecture that can utilize axonal delays- the DELTRON. We describe a network that modifies the delays associated with each afferent to memorize a spatio-temporal spike pattern. We describe our algorithm in the next section, followed by a possible hardware implementation of the same and finally conclude with the results.

Fig. 1: Delay-based learning model. N input neurons receive spikes from N synaptic afferents. Spike delays $\mathbf{d}=[d_1, d_2, d_3, \dots, d_N]$ are modified such that membrane potential $V(t)$ crosses the V_{thr} .

II. DELTRON: THEORY AND RESULTS

A. Algorithm

The DELTRON uses axonal conduction delays to learn spike patterns by modifying it using a rule similar to the one used by its weight modifying counterpart- the Tempotron [8]. Our model consists of N input neurons representing N synaptic afferents (Fig. 1). Each synaptic afferent receives spikes from the presynaptic neuron with different delays and generates exponentially decaying synaptic currents. The output neuron computes the membrane voltage $V(t)$ as a sum of the postsynaptic potentials (PSPs) $K(t)$ generated by all the incoming spikes.

$$V(t) = \sum_{t_i} K(t - t_i) \quad (1)$$

Here t_i is the time at which spike at the i th afferent fires. The PSP kernel is given by $K(t-t_i) = V_0(\exp[-(t-t_i)/\tau] - \exp[-(t-t_i)/\tau_s])$, where τ is the membrane time constant and τ_s the synaptic current time constant. The output of the neuron y is 1 if the membrane voltage crosses the threshold V_{thr} and the spike pattern is said to be learnt.

Using this delay-based learning, we determined the memory capacity of our model as the number of patterns that the model learns. The model is trained on P spike patterns. Each pattern consists of each afferent randomly firing once between 1 and 400 ms (x_i). Delays randomly chosen from a uniform distribution between 0 and 50 ms are assigned to the incoming spikes. Therefore we have a N -dimensional delay vector $\mathbf{d}$ as the modifiable parameter of the model. The membrane voltage is computed as the sum of kernel $K(t-t_i)$ where $t_i = x_i + d_i$. The learning algorithm consists of the following steps:

- 1) Delay vector $\mathbf{d}$ is initialized
- 2) In every iteration of the training process, a spike pattern is presented to the network
- 3) $V(t)$ is computed for each pattern. Time t_{max} at which $V(t)$ is maximum (V_{max}) is found
- 4) If $V_{max} \geq V_{thr}$, the spike pattern is learnt. Delays are not modified
- 5) If $V_{max} < V_{thr}$, $\Delta \mathbf{d}$ is calculated using gradient descent on the error function $E = V_{thr} - V_{max}$ according to:
$$\Delta \mathbf{d} = \sum_{t_i < t_{max}} K'(t_{max} - t_i)$$
where K' indicates the derivative of K .
- 6) Delays are modified according to $\mathbf{d} = \mathbf{d} + \eta \Delta \mathbf{d}$ if the mean value of V_{max} for all the patterns that are below V_{thr} has increased.
- 7) Learning consisting of steps 3-6 is stopped if any of the following conditions is encountered
 - a. All patterns are learnt
 - b. 5000 iterations are completed or
 - c. Change in V_{max} value is very small for 50 times
- 8) After training is completed, $V(t)$ is computed for all the patterns the network was trained on. If $V_{max} > V_{thr}$ for a pattern, output $y=1$ and that pattern has been learnt.

Intuitively, the DELTRON maximizes its memory capacity by only modifying those delay values which can be most effective in memorizing a pattern. Only those afferents which are producing a spike within the time window guided by K are hence modified. The parameters used for training are: $V_0=2.12$, $\tau=15$ ms, $\tau_s=\tau/4$, learning rate $\eta=5$ for the first 500 iterations and then reduced by 0.2 after every 500 iterations. V_{thr} is determined such that by chance a random spike pattern has very little probability of firing a neuron.

B. Simulation Results

We show in Fig. 2 an example of the operation of this algorithm. We generated $P=100$ random spike patterns in which each of the $N=100$ spike time is randomly chosen

Fig. 2: Probability distribution of V_{max} for a set of 100 random spike patterns before and after training showing a clear shift in the peak of the distribution for trained patterns.

between 1 and 400 ms. The blue solid curve shows the probability distribution of V_{max} values of the 100 spike patterns before the network is trained while the red curve with circles shows the distribution after training. There is a clear shift in the peak of the distribution indicating that the DELTRON is able to learn to respond preferentially to these spike patterns. The black dashed curve depicts the response of the network to a new random set of 100 patterns showing its response to new, not learnt patterns is relatively unchanged. The vertical dashed line shows a possible choice of V_{thr} to

Fig. 3: Memory capacity of the model. Different number of patterns were used to train the network. As the number of patterns used for training increases the fraction of patterns that the network learns reduces.

maximize response to learnt memories while ignoring others.

In the second experiment, we tested the network for its memory capacity. The generation of the random spike patterns was kept same as before. The network was then trained on $P=10, 20, 30, 50, 70$ and 100 patterns. The task of the model was to output if it has learnt the patterns. The mean value of the percentage of patterns that the network recalled (Fig. 3) is a function of the number of synapses. For 90% memory recall, this network can be trained on around 40 patterns. As the

number of training patterns increases beyond this, the two distributions shown in Fig. 2 become similar and merge.

We also tested the learning capacity of our model when noisy patterns were presented to it. The model was trained on random spike patterns as in the first task. Temporally jittered versions of spike patterns were generated by adding Gaussian noise to all the spike times. The ability of the model to recognize these jittered spike patterns for jitter=1.5 ms (Fig. 4)

Fig. 4: Memory capacity of the model in the presence of temporal noise. A jitter of 1.5 ms was added to all spike times. The ability of the model to recognize these noisy patterns was measured.

shows that only a few percentage of noisy patterns are not recalled by the network as compared to the unperturbed spike patterns. The model can tolerate a jitter of about 10% of τ .

III. HARDWARE ARCHITECTURE AND RESULTS

A. Architecture

We propose an efficient mixed-signal VLSI implementation of the DELTRON algorithm. Figure 5 depicts the system level view of our proposed system where a spiking sensor communicates patterns to be memorized by our network. The network uses digital tunable delay lines as the memory storage element and communicates output spikes to an analog chip that houses a synapse and a spiking neuron. The inter-module communication can be handled by the AER protocol [9]. During the learning process, another digital block is needed to estimate $t_{\max}$ from the output spike train of the neuron (the connection is indicated by a dashed line in the figure). It should be noted that since we want to estimate $V_{\max}$ from the output spike train, we have to ensure that the neuron generates spikes for any value of $V_{\max}$. Hence, during the learning phase, the value of the threshold of the integrate and fire neuron is kept at zero. An advantage of this algorithm is that synaptic mismatch and DACs to implement weights are avoided. All the digital blocks, described in the following paragraphs, are also relatively simple.

- **Time-keeper:** A global counter is kept which counts up till the maximum value of pattern time which is 400 ms in this case. The corresponding digital word

will be referred to as T_s and indicates the system

Fig. 5: System architecture of the proposed hardware implementation of the DELTRON algorithm uses digital for learning and analog for computation.

time.

- **Delay line:** Every axon has two registers, R1 and R2. While R1 stores the current delay value 'd', R2 stores the value of $(x+d)$ which is the time when it should generate a spike. Whenever there is an incoming spike, R2 will be updated to the sum of d and T_s , which equals x at that time instant. At every clock the value in R2 is compared with the current time T_s , and if found same, the axon fires an output event request to the AER_{out} module.
- **Estimate $t_{\max}$:** This block comprises two counters, C1 and C2, and two registers to store the current estimate of $V_{\max}$ and $t_{\max}$ respectively. We estimate $V_{\max}$ by measuring the inter-spike intervals (ISI)—a large $V_{\max}$ corresponds to a small ISI and vice versa. At the beginning of every cycle, C1, C2 and the $t_{\max}$ register are reset to zero while the ISI register is reset to the maximum time. Every odd spike resets and starts C1 while stopping C2, while every even spike does the opposite. The counter that is stopped holds the current ISI which is compared with the current *minimum* ISI stored in the ISI register. If the current ISI is smaller, it is stored in the ISI register while $t_{\max}$ is updated as $T_s - \text{ISI}/2$.
- **Learning module:** After every input pattern presentation is over, the learning module will update the delays before the presentation of the next pattern. First, every R2 register is updated to the difference of $t_{\max}$ and its current value (computing $t_{\max} - t_i$ in step 5 of the algorithm in Sec IIA). Only those axons with a positive value in R2 after this step get updated. The update is done serially by using the value in R2 of a chosen axon to index a look-up table that stores the values of $\eta K'(t)$. The value in R1 is modified by adding the value from the LUT to its current value.

B. Results

As mentioned earlier, an important step in the algorithm is the estimation of $V_{\max}$ from the spike train output of an integrate and fire neuron. We have performed SPICE simulations of this part of the algorithm using transistor models from the AMS 0.35 μ m CMOS process. We used an approximation to a conductance based neuron model [10] for this simulation. We modified the original structure by

Fig. 6: Current input waveform to the integrate and fire neuron and ISI of the output spike train show that higher the current, smaller is the ISI and vice versa. Absolute values are not shown for ease of viewing.

including an OTA as an explicit comparator setting a well-defined value of V_{thr} . The synapse model used in the simulation is based on the differential pair integrator structure described in [11].

Figure 6 plots an example of the input current waveform for an input spike train consisting of 10 spikes at random times. The output spikes from the neuron can be seen to provide a good estimate of the input current—larger the input current,

Fig. 7: A plot of the estimated and actual $t_{\max}$ values exhibit a high degree of correlation.

smaller is the ISI or equivalently higher is the density of spikes. This validates our earlier assumption that ISI can be used to estimate $V_{\max}$ (in this case it is equivalent to $I_{\text{in},\max}$). Furthermore, Fig. 7 plots the relationship between t_{est} (estimated $t_{\max}$) and t_{act} (actual $t_{\max}$) for 10 different random

spike trains where the input consists of 10 spikes at random times between 0 and 100ms. The high degree of correlation of these two values again validate the choice of ISI as a metric for $I_{\text{in},\max}$. As mentioned earlier, the threshold was kept at a very small value for this part. It should be noted that in this case I_{leak} is the effective threshold since we are sensing the maximum of the input current.

IV. DISCUSSIONS AND CONCLUSIONS

In this paper, we presented a new spiking neural network that modifies axonal delays to learn spatio-temporal spike patterns. We use a learning rule similar to the Tempotron but modify delays instead of weights. An efficient hardware mapping of this algorithm is also presented which can share one analog synapse and neuron with programmable delay connections in the digital domain.

A major difference of this network compared to weight modifying ones is that weight modifications can increase the $V_{\max}$ value without bound. However, delay modifications can only increase $V_{\max}$ in a limited way, the maximum being equal to the sum of the EPSP amplitudes of all afferents. Hence, it is difficult to get good pattern separability by this approach alone. An improvement of the present method might be to use a voting strategy to combine the outputs of multiple such networks trained in parallel. We are currently exploring this possibility.

ACKNOWLEDGEMENTS

The authors would like to thank Shoushun Chen, Zohair Ahmed, Roshan Gopalakrishnan and Subhrajit Roy for useful discussions and help with SPICE simulations.

REFERENCES

- [1] V. Chan, S. C. Liu and A. Van Schaik, "AER EAR: A Matched Silicon Cochlea Pair with Address Event Representation Interface," *IEEE Trans. Circuits And Systems-I*, vol. 54, no. 1, pp. 48-59, 2007.
- [2] P. Lichtsteiner, C. Posch and T. Delbruck, "A 128X128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566-76, 2008.
- [3] G. Indiveri et al., "Neuromorphic Silicon Neuron Circuits," *Frontiers in Neuroscience*, vol. 5, 2011, doi : 10.3389/fnins.2011.00073.
- [4] C. Bartolozzi and G. Indiveri, "Synaptic Dynamics in Analog VLSI," *Neural Computation*, vol. 19, no. 10, pp. 2581-2603, 2007.
- [5] A. Basu, S. Ramakrishnan, C. Petre, S. Koziol S. Brink and P. Hasler, "Neural Dynamics in Reconfigurable Silicon," *IEEE Trans. on Biomedical Circuits and Systems*, vol. 4, no. 5, pp. 311-19, Oct. 2010.
- [6] E. Izhikevich, "Polychronization: Computation with Spikes," *Neural Computation*, vol. 18, pp. 245-282, 2006.
- [7] R. Wang, C. Jin, A. McEwan and A. VanSchaik, "A programmable axonal propagation delay circuit for time-delay spiking neural networks," in *Proc. IEEE ISCAS*, pp. 869-72, 2011.
- [8] R. Gutig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing based decisions," *Nature Neuroscience*, vol. 9, pp. 420-428, 2006.
- [9] M. Mahowald, "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function," Ph.D. Dissertation, California Inst. of Tech., 1992.
- [10] P. Livi and G. Indiveri, "A current-mode conductance-based silicon neuron for Address-Event neuromorphic systems," in *Proc. of IEEE Intl. Symp. Circ. and Sys.*, pp. 2898-2901, 2009.
- [11] C. Bartolozzi and G. Indiveri, "Synaptic dynamics in analog VLSI," *Neural Computation*, vol. 19, no. 10, pp. 2581-2603, 2007.