Implementing Spike-Timing-Dependent Plasticity on SpiNNaker Neuromorphic Hardware

Xin Jin, Alexander Rast, Francesco Galluppi, Sergio Davies, and Steve Furber

***Abstract*— This paper presents an efficient approach for implementing spike-timing-dependent plasticity (STDP) on the SpiNNaker neuromorphic hardware. The event-address mapping and the distributed synaptic weight storage schemes used in parallel neuromorphic hardware such as SpiNNaker make the conventional pre-post-sensitive scheme of STDP implementation inefficient, since STDP is triggered when either a pre- or post-synaptic neuron fires. An alternative pre-sensitive scheme approach is presented to solve this problem, where STDP is triggered only when a pre-synaptic neuron fires. An associated deferred event-driven model is developed to enable the pre-sensitive scheme by deferring the STDP process until there are sufficient history spike timing records. The paper gives detailed description of the implementation as well as performance estimation of STDP on multi-chip SpiNNaker machine, along with the discussion on some issues related to efficient STDP implementation on a parallel neuromorphic hardware.**

I. Introduction

Synaptic plasticity is one of the most important features of a neural network, and many different plasticity mechanisms have been developed since the last century to mimic the biological process of learning. Spike-timing-dependent plasticity (STDP) based on Hebbian theory has received much attention in recent years [4], [13].

In this paper the approach towards developing STDP rule on SpiNNaker is demonstrated. The STDP rule modifies synaptic weights according to the difference between pre- and post-synaptic spike timings. The ordering decides whether the modification is potentiation or depression. Normally, STDP is triggered whenever a pre-synaptic spike arrives, or a post-synaptic neuron fires, which in turn requires keeping indices of synaptic information in both pre- and post-synaptic orders for efficiency [4]. The difficulty in implementing this scheme on SpiNNaker's event-address mapping (EAM) model is that when a post-synaptic neuron fires, relevant synaptic weights are still located in the external memory. Synaptic information will show up in the local memory only when a pre-synaptic spike arrives.

This problem is solved by applying a novel pre-synaptic sensitive scheme with an associated deferred event-driven model. The pre-sensitive scheme only triggers the STDP when a pre-synaptic spike arrives and requires keeping only one index of synaptic weights (in pre-synaptic order), hence it reduces the processing and the memory bandwidth requirements. However, the pre-synaptic sensitive scheme relies on "future" spike timing information to perform STDP. The deferred event-driven model postpones the STDP for a certain period of time, waiting for the emergence of future spike timing, enabling then the pre-synaptic scheme.

The authors are with the School of Computer Science, The University of Manchester, Manchester, UK (email: {jinxa}@cs.man.ac.uk)

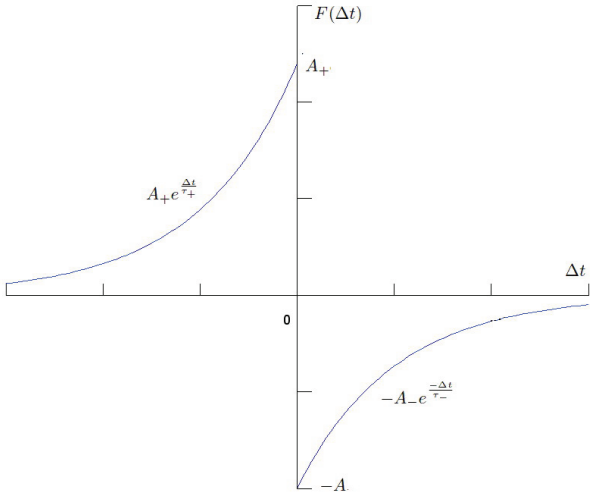


Fig. 1. The STDP modification function.

The pre-sensitive scheme and deferred event-driven model proposed are verified and evaluated by running a four-chip SpiNNaker simulation and comparing the results with the Matlab simulation using pre-post-sensitive scheme. When enabling STDP on SpiNNaker, this scheme does involve certain approximation however, as compared to the conventional STDP implementation. The impact of the approximation on the system performance will be discussed at the end of this paper.

The rest of the paper is organized as following: Section II gives a brief introduction to the STDP rule. Section III deals with the difficulties of implementing STDP on SpiNNaker, presenting the pre-sensitive scheme and the deferred event-driven model to tackle these difficulties; finally the algorithm is converted into the actual implementation. Section IV presents some simulations along with results. Related issues are discussed in Section V followed by a conclusion in Section VI.

II. STDP

The learning rule implemented on SpiNNaker is the well-known spike-timing-dependent plasticity (STDP) [4], [13], where the amount of weight modification is decided by the function shown below:

$$F(\Delta t) = \begin{cases} A_+ e^{\frac{\Delta t}{\tau_+}} & \Delta t < 0, \\ -A_- e^{\frac{-\Delta t}{\tau_-}} & \Delta t \geq 0. \end{cases} \quad (1)$$

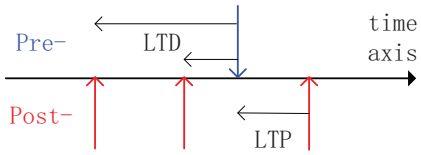


Fig. 2. The pre-post-sensitive scheme. STDP is triggered when either a pre-synaptic neuron fires or a post-synaptic neurons fires.

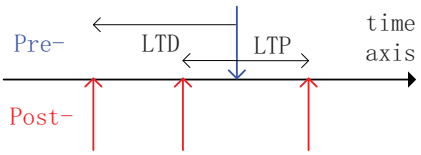


Fig. 3. The pre-sensitive scheme. STDP is triggered only when a pre-synaptic neurons fires (a spike arrives).

Where Δt is the time difference between the pre- and post-synaptic spike timing ($\Delta t = T_{pre} - T_{post}$, being T_{pre} the pre-synaptic spike time stamp and T_{post} the post-synaptic spike time stamp), A_+ and A_- are the maximum synaptic modifications, and τ_+ and τ_- are the time windows determining the range of spike interval over which the STDP occurs. If the pre-synaptic spike arrives before the post-synaptic neuron fires (i.e. $\Delta t < 0$), it causes long-term potentiation (LTP) and the synaptic weight is strengthened according to $A_+ e^{\frac{\Delta t}{\tau_+}}$. If the pre-synaptic spike arrives after the post-synaptic neuron fires (i.e. $\Delta t \geq 0$), it causes long-term depression (LTD) and the synaptic weight is weakened according to $A_- e^{\frac{-\Delta t}{\tau_-}}$. The corresponding function curve is shown in Figure 1. The choice of STDP parameters can be found elsewhere [13].

III. Methodology

A. The pre-post-sensitive scheme

In most desktop computer simulations, the implementation of STDP is quite straightforward. Because all synaptic weights are locally accessible, the STDP can be triggered whenever a spike is received or a neuron fires. In this approach, calculating Δt is simply a matter of comparing the history records of spike timings. This corresponds to examining the past spike history (at least within the STDP sensitivity window), as shown in Figure 2 where pre-synaptic spikes are shown in blue and post-synaptic spikes are shown in red. We call the STDP triggered by the receiving spike as a *pre-synaptic STDP*, since it is caused by a pre-synaptic spike; and the STDP triggered by a neuron firing hence as a *post-synaptic STDP*, since it is caused by a post-synaptic spike. The pre-synaptic STDP causes LTD which depresses the connection, whereas the post-synaptic STDP causes LTP which potentiates the connection. The scheme used by most desktop computer simulations is termed a *pre-post-sensitive scheme*, since both pre-synaptic and post-synaptic STDPs are involved.

Problems on SpiNNaker: SpiNNaker uses a distributed memory system. Each chip is associated with one SDRAM shared by 20 processors. Each processor has a fast internal memory called DTCM. According to the Event Address Mapping (EAM) scheme, synaptic weights are pre-synaptic indexed and stored in the SDRAM of the post-synaptic end. They will only be transmitted to the DTCM by a DMA operation when a spike arrives. This memory system guarantees a good balance between the memory space and the accessing speed. Detailed description of SpiNNaker can be found in [6], [8], [7], [12], [11].

Two problems arises however if STDP is implemented using the conventional pre-post-sensitive scheme:

1) The required synaptic weights are NOT in the DTCM when a local neuron fires which disables post-synaptic STDP. It is inefficient to use a second DMA operation to move synaptic weights from the SDRAM to the DTCM when a neuron fires, as it will double the memory bandwidth requirement.
2) Since the synapse block is a neuron-associative memory array, it can only be indexed either by the pre- or post-synaptic neuron. If synapses are stored in pre-synaptic order, the pre-synaptic STDP will be very efficient while the post-synaptic STDP will be inefficient, and vice versa - because one or the other lookup would require a scattered traverse of discontiguous areas of the synaptic block.

As a result, an alternative scheme is required for STDP implementation on SpiNNaker.

B. The pre-sensitive scheme

We propose a new scheme for implementing STDP on SpiNNaker, called the *pre-sensitive scheme* as shown in Figure 3. The pre-sensitive scheme triggers both pre-synaptic STDP (LTD, left headed arrow) and post-synaptic STDP (LTP, right headed arrow), when a pre-synaptic spike arrives. This ensures the synaptic weights are always in the internal DTCM when STDP is triggered, and makes accessing individual synapses possible by efficient iteration through the array elements when the synapse block is in pre-synaptic order.

The difficulties: However the implementation of the pre-sensitive scheme is not as easy as the pre-post-sensitive scheme. There are two difficulties involved:

1) This scheme requires the examination of not only the past spike history records, but also of future records. Naturally, future spike timing information is not available at the time the pre-synaptic spike arrives since it has not yet happened.
2) SpiNNaker supports a range of synaptic delays from 0 ms to 15 ms for each connection [5] to compensate for the time differences between electronic and neural timings. The spike arrives at the electronic time rather than the neural time, while the effect of the input is deferred until its neural timing due to the delay. The STDP should be started at the neural time.

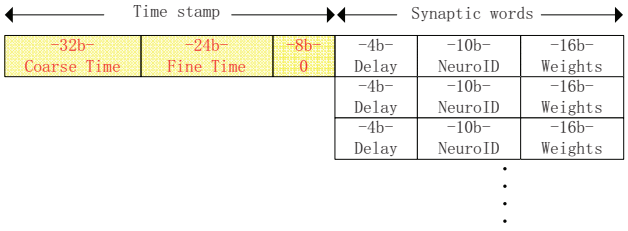


Fig. 4. The pre-synaptic time stamp.

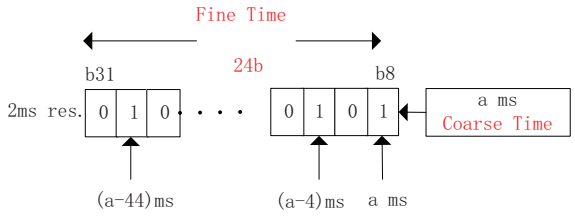


Fig. 5. The time stamp representation.

C. The deferred event-driven model

Both of the above difficulties can be overcome by deferring the STDP operation by introducing another model termed *deferred event-driven* (DED) model. In the DED model, no STDP is triggered immediately on receiving a pre-synaptic spike. Instead, the spike timing is recorded as a time stamp and STDP is triggered after waiting a certain amount of time (the current time plus the maximum delay and the time window). The DED model ensures that information on future spike timings is obtained.

1) Timing records: STDP requires information on both pre-synaptic and post-synaptic spike timings. A pre-synaptic time stamp at 2ms resolution is kept in the SDRAM along

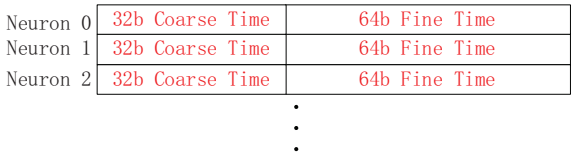


Fig. 6. The post-synaptic time stamp.

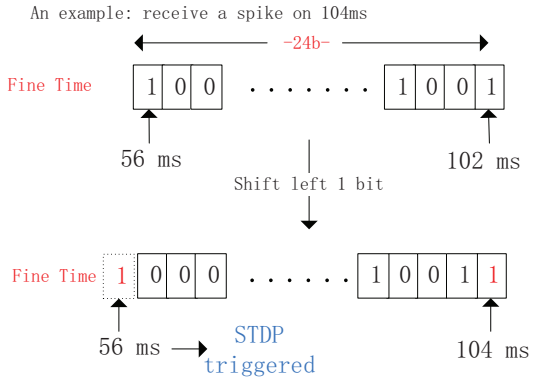


Fig. 7. Updating the pre-synaptic time stamp.

with each synapse block as shown in Figure 4 (the global ID of the pre-synaptic neuron is added in front of the time stamp for debugging purposes), and is updated when pre-synaptic spikes arrive. The time stamp comprises two parts, a coarse and a fine time. The coarse time is a 32-bit digital value representing the last time the neuron fired. The fine time is a bit-mapped field of 24 bits (bit [31:8]) representing spike history in the last 48 ms. The coarse time always points to the least significant bit of the fine time (bit 8). As a result, the least significant bit (bit 8) of the fine time is always set.

Figure 5 shows how time history is represented by the time stamps. Assuming the coarse time is a ms, bit 8 in the fine time represents the last spike arriving at a ms. Each higher bit represents a spike arrival time which is 2 ms earlier. In Figure 5 for instance, it is able to calculate that the pre-synaptic spikes arrive at a, $(a-4)$ and $(a-44)$ ms respectively.

Post-synaptic time stamps reside in local DTCM (Figure 6) and have a similar format to pre-synaptic time stamps except that they are 64 bits long (bit [63:0], representing 128ms), allowing longer history records.

2) Updating timing records: A pre-synaptic time stamp is updated when a packet is received. During the update, firstly, the coarse time is subtracted from the new time to produce a time difference t_{dif}, as shown in Figure 7. The time difference is divided by the time resolution, to get the number of bits to be shifted (2ms in this case, so the shift is by $t_{dif}/2$ bits). Then the fine bit is shifted left by $t_{dif}/2$ bits. If any "1" is shifted out of the most significant bit, STDP will be triggered. Bit 32 represents the pre-synaptic spike time which triggers STDP.

The updating of the post-synaptic time stamp is similar to that for the pre-synaptic, except:

1) The post-synaptic time stamp is updated when a neuron fires.
2) The update of the post-synaptic time stamp will NOT trigger STDP.

D. The STDP process

If STDP is triggered by a "1" going to bit 32 in the pre-synaptic fine time, its post-synaptic connections in the Synaptic Block are firstly traversed word by word. For each Synaptic Word (one connection), the pre-synaptic spike time (the time of bit 32) is added to the synaptic delay to convert the electronic timing to the neural timing T; the processor then calculates the LTD $[T-\tau_-, T]$ and the LTP $[T, T+\tau_+]$ windows. If any bit in the post-synaptic time stamp is set within the LTD or LTP window, the synaptic weight is either weakened or strengthened according to the STDP rule.

The post-synaptic time stamp can be retrieved from the DTCM as determined by the neuron ID in the Synaptic Word. The processor then scans the post-synaptic time stamp looking for any "1" located within the learning window, and updates the weight accordingly.

E. Implementation

The flow chart of the STDP implementation is shown in Figure 8 which comprises three nested loops in the

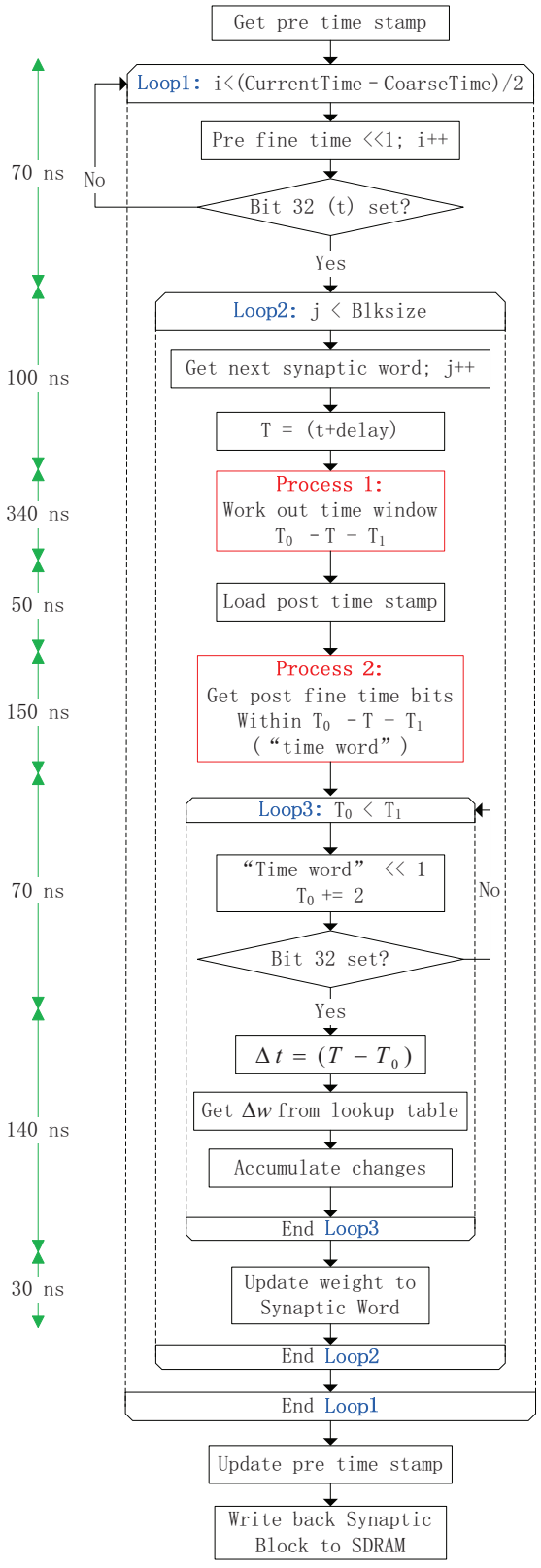


Fig. 8. STDP implementation flow chart.

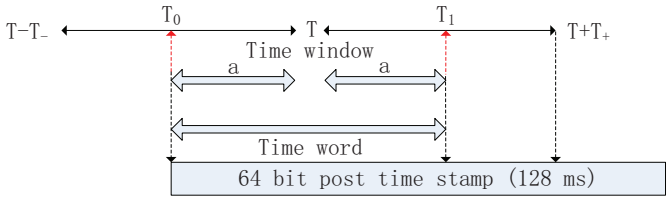


Fig. 9. Calculate the time window and time word.

programme to handle a new spike and to do STDP. Each of the three loops may run through several iterations:

- Loop1: Update the pre-synaptic time stamp. This loops $t_{dif}/2$ times. If there are n "1"s shifted to bit 32, the STDP will be triggered n times.
- Loop2: Traverse the post-synaptic connections. This loops $m = Blksize$ times, the number of words in a Synaptic Block, i.e. the number of post-synaptic connections in this fascicle from the pre-synaptic neuron that fired.
- Loop3: Scan the post-synaptic time stamp. This loops $x = (T_1 - T_0)/2$ times (T_0 and T_1 will be explained later), and each time 1 bit will be detected. If there are y bits found within the time window, the weight updating will be executed y times.

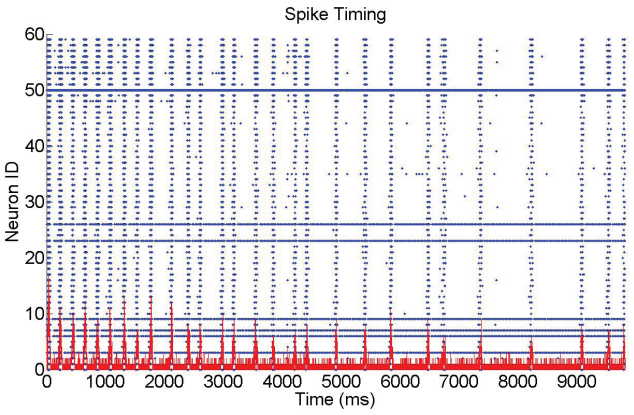
The computational complexity of the bit-detection in Loop3 is $o(nmx)$ and the computational complexity of weight updating in Loop3 is $O(nmy)$. As a result the shifting and the weight updating in Loop3 needs to be as efficient as possible.

Here follow a detailed description of the main processes used in the algorithm:

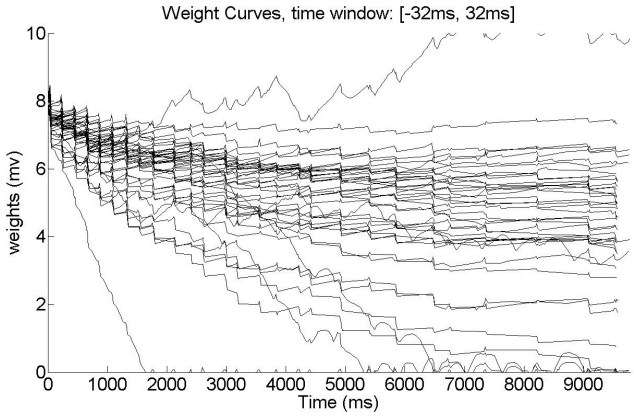
1) Process 1 - the time window: Process 1 in flow chart 8 is responsible for calculating the time window, in this implementation a dynamic window, from T_0 to T_1, which differs from the window defined in the STDP rule by τ_- and τ_+. Three restrictions are applied when calculating the time window (shown in Figure 9):

1) The time window must be in the range of $[\tau_-, \tau_+]$.
2) There are history records in the post-synaptic time stamp in the time window. In Figure 9, the time window becomes $[T_0, T + T_+]$.
3) The left window and the right window are the same length. In Figure 9, the time window becomes $[T_0, T_1]$, as $T - T_0 = T_1 - T = a$.

2) Process 2 - the time word: The post-synaptic fine time stamp field is 64 bits and a 64-bit shifting operation in ARM takes 8-9 CPU cycles, while a 32-bit one takes only 1. The meaningful bits of the fine time stamp are those within the time window $[T_0, T_1]$, which is smaller than 32 bits if $\tau_- = \tau_+ \leq 32ms$. These bits are referred to as the "time word", which represents the bits of the post-synaptic fine time stamp within the time window $[T_0, T_1]$, after bit-time conversion. If any of the bits is set in the "time word", the weight needs to be updated accordingly.

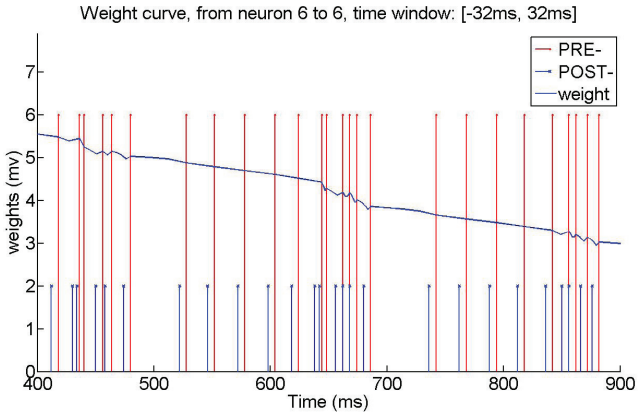


(a) Spike raster plot.

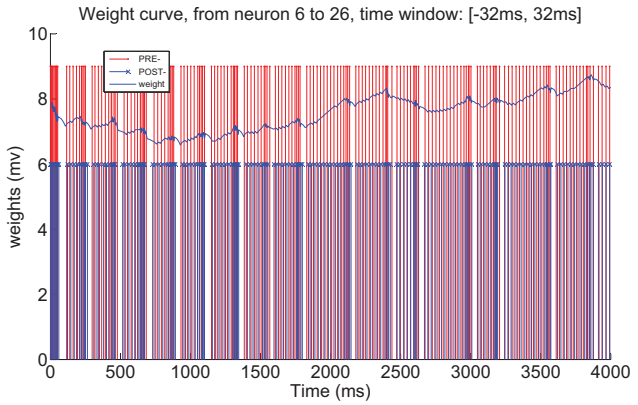


(b) Weight curves of connections from pre-synaptic neuron 6. The synaptic weight going rapidly to 0 is a self-connection.

Fig. 10. STDP results.



(a) Synapse of neuron 6 to 6 (weaken) within 400 – 900 ms.



(b) Connection from neuron 6 to 26 (strengthen) within 0 – 4000 ms.

Fig. 11. Weight modification caused by the correlation of the pre and post spike times.

3) Bit detection: There are two bit detection operations in the STDP implementation. The "LSLS" instruction provided by the ARM instruction set is efficient in detecting if there is any "1" moved into the carry bits (bit 32), and allows the processor to do a conditional branch. To use the "LSLS" instruction, bits [31:8] (instead of bit [23:0]) of a word is used for the pre-synaptic fine time stamp.

4) Lookup table: Since the parameters of the STDP are determined before the simulation starts, the Δw can be pre-computed based on different values of Δt and loaded into a lookup table. When the Δt is obtained, Δw can easily be fetched from the lookup table. Compared to the real-time computation of Δw, using a lookup table is obviously more efficient.

5) Performance: The processor time usage for each step of processing is shown in Figure 8 where process 1 and 2 in Loop 2 are the most time consuming operations. The calculation of Δw in Loop 3 takes only 140 ns, with the help of a lookup table.

IV. Simulation Results

A. 10s 60-neuron test

A 60-neuron network is simulated on the four-chip SpiNNaker SOC Designer model. The network is largely based on the code published in [4] (but in a smaller scale), which was also used to test the consistency of our results. There are 48 Regular Spiking excitatory neurons $a = 0.02, b = 0.2, c = -65, d = 8$ and 12 Fast Spiking inhibitory neurons $a = 0.1, b = 0.2, c = -65, d = 2$. Each neuron connects randomly to 40 neurons (self-connections are possible) with random 1-15 ms delays; inhibitory neurons only connect to excitatory neurons. Initial weights are 8 and -4 for excitatory and inhibitory connections respectively. Parameters $\tau_+ = \tau_- = 32ms$, $A_+ = A_- = 0.1$ are used for STDP. Inhibitory connections are not plastic [1]. Following learning the weights of excitatory neurons are clipped to [0, 20]. There are 6 excitatory and 1 inhibitory input neurons, receiving constant input current $I = 20$ to maintain a high firing rate. Weights are updated in real-time (every 1 ms).

The simulation is run for 10 sec (biological time) and Figure 10(a) shows the spike raster, and Figure 10(b) shows the evolution of synaptic weights of connections from pre-synaptic neuron ID 6 (an input neuron). At the beginning of

the simulation input neurons fire synchronously, exciting the network which exhibits high-amplitude synchronized rhythmic activity around 5 to 6 Hz. As synaptic connections evolve according to STDP, uncorrelated synapses are depressed while correlated synapses are potentiated. Since the network is small and the firing rate is low, most synapses will be depressed, leading to a lower firing rate. The synaptic weight going rapidly to zero is the self-connection of neuron ID 6: since each pre-synaptic spike arrives shortly after the post-synaptic spike the synapse is quickly depressed.

Detailed modifications of the self-connection weight (the blue curve) is shown in Figure 11(a) along with pre- (red vertical lines) and post-synaptic timing (blue vertical lines), from 400 ms to 900 ms. Modification is triggered by pre-synaptic spikes. The weight curve between two pre-synaptic spikes is firstly depressed because of the LTD window and then potentiated because of the LTP window. The detailed modification of the strengthened synapse (from neuron 6 to 26) from 0 ms to 4000 ms is shown in Figure 11(b).

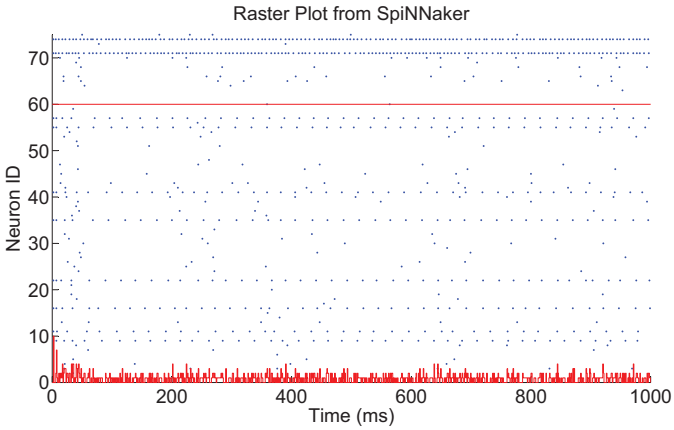
B. 30s 76-neuron test

The system is also tested by a 30-second simulation of a 76-neuron network (60 excitatory and 16 inhibitory neurons) with each excitatory neuron randomly connects to 10 other neurons and each inhibitory neurons randomly connects to 10 excitatory neurons. A random 10 neurons receive a constant biased input of 20 mV. A comparison between a simulation without STDP (Figure 12(a)) and with STDP (Figure 12(b)) after the first 1,000 ms, is shown. The simulation with STDP shows a more synchronized firing pattern than the simulation without.

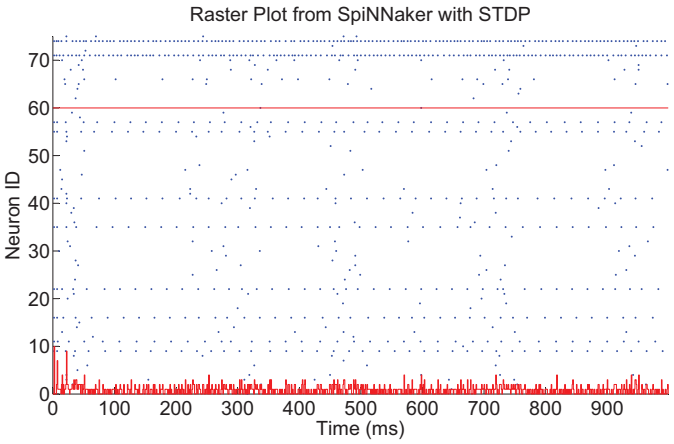
The result from the 30th second is shown in Figure 12(c), and the corresponding Matlab (fixed-point, using pre-post sensitive scheme) results[1] is shown in Figure 13. The firing pattern during the 30th second shows a more obviously synchronized behavior. Figure 14(a) shows the activity of neuron ID 1 during the 1st second. Figure 14(b) shows the activity of neuron ID 1 during the 30th second.

The weight distribution at the end of the simulation can be observed in Figure 15: most of the excitatory weights are slightly de/potentiated around their initial value of 10. Some connections are potentiated up to their maximum (20) because they are systematically reinforced, due to converging delays. For example we found a group of 7 neurons strongly interconnected at the end of the simulation. Examining their connections and delays (as shown in Figure 16 two circuits with converging delays could be found: the first one starts from neuron 57 and propagates through neuron 8, 47 and 43, ending at neuron 19; the second one starts from neuron 55 exciting neuron 57, propagates through neurons 47 and 19 and ends at neuron 42. All the weights are strongly potentiated, near or up to their maximum at the end.

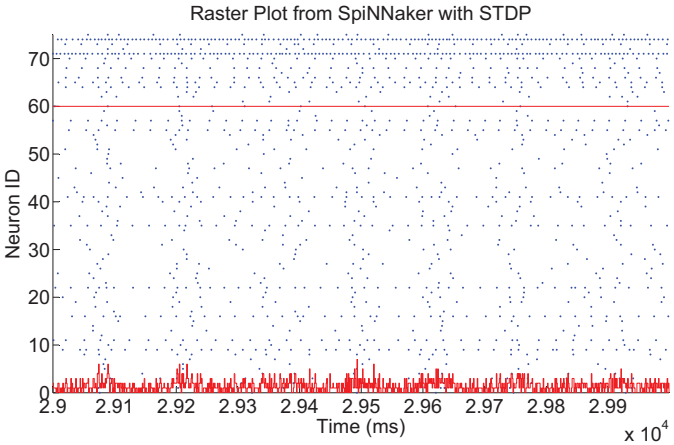
[1]The Matlab simulation is based on the code provided in [4]. The STDP parameter setting is slight different: $\tau_+ = \tau_- = 15ms$, $A_+ = 0.1$, and $A_- = 0.12$; weights are updated every 1 second instead of 1 millisecond.



(a) Simulation with STDP disabled during the 1st second.



(b) Simulation with STDP enabled during the 1st second.



(c) Simulation with STDP enabled during the 30th second.

Fig. 12. Comparison between the simulation with and without STDP on SpiNNaker during 1st second and 30th second.

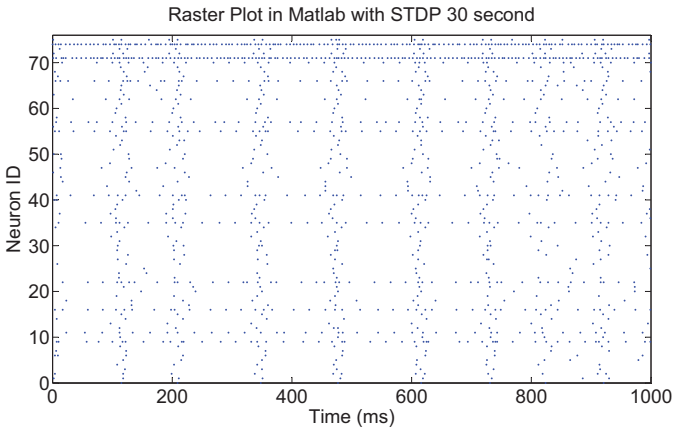
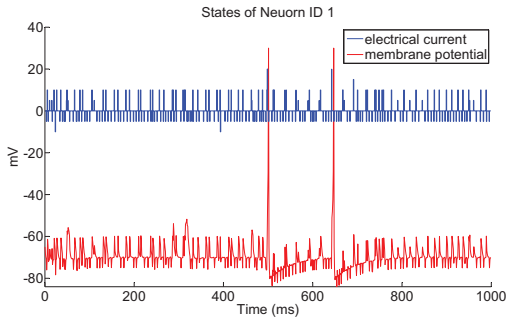
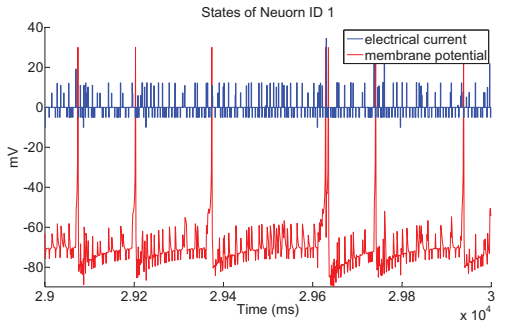


Fig. 13. The STDP result from the Matlab simulation (fixed-point) with parameters: $\tau_+ = \tau_- = 15ms$, $A_+ = 0.1$, and $A_- = 0.12$.



(a) The activity of neuron 1 during the 1st second.



(b) The activity of neuron 1 during the 30th second.

Fig. 14. The behavior of an individual neuron (ID 1) during the 1st second and the 30th second.

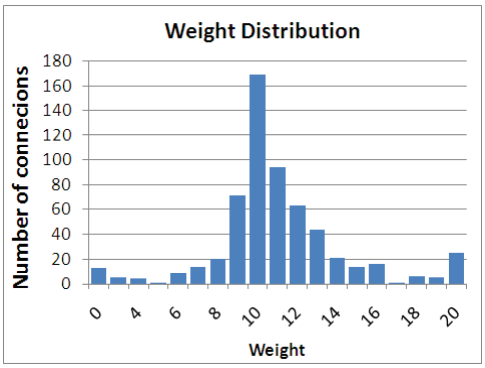


Fig. 15. Weight distribution at the end of the simulation. Weights for excitatory neurons are clipped to [0, 20]

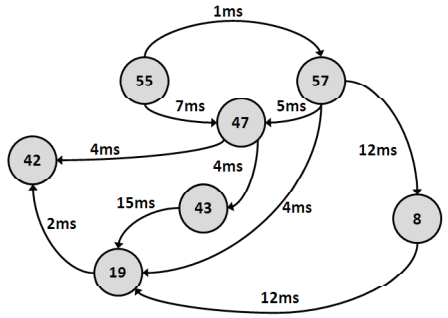


Fig. 16. Group of neurons with converging delays. It is possible to track down two circuits with converging delay (neurons 55, 57, 47, 19 ending at neuron 42 and neuron 57, 47, 43 and 8 ending at neuron 19.

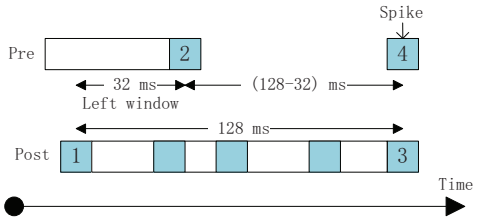


Fig. 17. Relationship between the firing rate and the length of timing records.

V. Discussion

A. Firing rates and the length of timing records

The length of the time stamp affects both the performance and precision of the STDP rule. Longer history records yield better precision at the cost of significantly increased computation time. The optimal history length is therefore dependent upon the required precision and performance. A 24-bit pre-synaptic time stamp with 2 ms resolution and a maximum of 15 ms delay guarantees a $24 * 2 - 15 > 32ms$ right (LTP) window for any delay.

The pre-sensitive scheme and the deferred event-driven model require the new input to push the old input record into the carry bit to trigger STDP. What happens, however, if the new input does not come or it comes at a very low rate? The post-synaptic time stamp is pushed forward when new post-synaptic spikes are generated, and the history records will be lost. If the pre-synaptic firing rate is too low, there will be no post-synaptic time records within the time window at the time STDP is triggered. As a result, there are certain restriction in terms of the firing rate of pre-synaptic neurons to ensure that STDP will be triggered in time. As shown in Figure 17, at the time a new pre-synaptic spike (4) arrives, the time difference between pre-synaptic spike 2 and post-synaptic spike 1 is 32 ms – the same size of left window (LTD) as the size of right window. Let the average interval of two pre-synaptic spikes to be T_{pre}, and the average interval of two post-synaptic spikes to be T_{post}.

1) When $T_{post} \leq T_{pre} + 32$ (post-synaptic neurons fire more frequently), to guarantee a 32 ms left window, the interval between two pre-synaptic spikes must be

no more than $128 - 32 = 96ms$; this, in turn, requires a firing rate of more than $1000/96 = 10.4Hz$.

2) When $T_{post} > T_{pre} + 32$ (post-synaptic neurons fire less frequently), a 32 ms left window can be guaranteed with any pre-synaptic firing rate.

B. Approximation and optimization

Since the Matlab and SpiNNaker simulations employ different implementation schemes, exactly the same results are not achievable. At present the role of different synaptic types is imperfectly understood and remains an area of intense research interest [2]. Equally significantly, the level of biophysical realism necessary to achieve useful behavior or model actual brain dynamics is unclear. For instance, in the case of the well-known STDP plasticity rule, while many models exist describing the behavior [3], the actual biological data regarding STDP is noisy and of low accuracy. Observed STDP synaptic modifications exhibit a broad distribution for which the nominal functional form of STDP models usually constitute an envelope or upper bound to the maximum modification [1], [9]. This suggests that high repeatability or precision in STDP models is not particularly important.

While SpiNNaker is capable of modelling synapses with biophysical realism down to the molecular level if necessary, such high biological fidelity is computationally expensive. For understanding the computational properties of synapses, such exact replication appears to be unnecessary in view of the observed synaptic variability. Equally, however, fidelity to a precise functional form need not be particularly high. This gives considerable latitude for experimenting with different synaptic models in order to investigate various tradeoffs between computational cost and functional accuracy.

The SpiNNaker STDP implementation can be further simplified by using "nearest spike approximation" [10] which limits LTD/LTP to the first/last presynaptic spike before/after the postsynaptic one. The implementation of the STDP rule involves a series of processing steps. Most of the processing steps are in nested loops and will be executed for a number of iterations during the STDP process. Thus the performance will decrease significantly with STDP enabled, a common problem of using STDP. The use of the "nearest spike approximation" potentially reduces the number of iterations, and will therefore significantly reduce the overhead.

The length and resolution of the time stamp are reconfigurable to meet different requirements; if a larger time window is required, the length of the pre-synaptic time stamp can be increased or the resolution can be reduced to 4 ms per bit. The dynamic adjustment of the time stamp length is also a possible optimization, by which users will be able to modify the length of the time stamp or the time resolution at run-time, to meet the accuracy-performance requirement during different simulation periods.

VI. Conclusion

In this paper we have presented a way to implement an efficient STDP algorithm on the SpiNNaker, a multi-chip neuromorphic parallel hardware with a distributed memory system, where synaptic data is stored on the post-synaptic end and is retrieved only upon the arrival of pre-synaptic spikes. The problem is solved by introducing a pre-synaptic scheme and an associated deferred event-driven model, which permits to update the weights only when a pre-synaptic spike arrives, hence reducing memory requests and indexing process operations. The methods shown in this paper validate the practicality of learning on multi-chip neuromorphic parallel hardware and illustrates ways to translate theoretical learning rules into actual implementation, leading to the further development of universal learning support for spiking neural networks on neuromorphic hardware.

Acknowledgment

We would like to thank the Engineering and Physical Sciences Research Council (EPSRC), Silistix, and ARM for support of this research.

References

[1] Guoqiang Bi and Muming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, 18(24):10464–10472, 1998.

[2] D. Durstewitz. Implications of synaptic biophysics for recurrent network dynamics and active memory. 22(8):1189–1200, October 2009.

[3] M. Hartley, N. Taylor, and J. Taylor. Understanding spike-time-dependent plasticity: A biologically motivated computational model. *Neurocomputing*, 69(16):2005–2016, July 2006.

[4] Eugene M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18(2):245–282, February 2006.

[5] X. Jin, S. Furber, and J. Woods. Efficient modelling of spiking neural networks on a scalable chip multiprocessor. In *Proc. 2008 International Joint Conference on Neural Networks*, Hong Kong, 2008. inproceedings.

[6] X. Jin, F. Galluppi, C. Patterson, A.D. Rast, S. Davies, S. Temple, and S.B. Furber. Algorithm and software for simulation of spiking neural networks on the multi-chip spinnaker system. In *Proc. 2010 International Joint Conference on Neural Networks*, 2010.

[7] X. Jin, M. Lujan, M.M. Khan, L.A. Plana, A.D. Rast, S.R.Welbourne, and S.B. Furber. Algorithm for mapping multilayer bp networks onto the spinnaker neuromorphic hardware. In *Proc. International Symposium on Parallel and Distributed Computing (ISPDC'2010)*, 2010.

[8] Xin Jin. *Parallel Simulation of Neural Networks on SpiNNaker Universal Neuromorphic Hardware*. PhD thesis, Computer Science, The University of Manchester, 2010.

[9] H. Markram and M. Tsodyks. Redistribution of synaptic efficacy between neocortical pyramidal neurons. *Nature*, (382):807–810, 1996.

[10] Timothe Masquelier, Rudy Guyonneau, and Simon J. Thorpe. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE*, 3(1):e1377, 01 2008.

[11] A. D. Rast, F. Galluppi, X. Jin, and S.B. Furber. The leaky integrate-and-fire neuron: A platform for synaptic model exploration on the spinnaker chip. In *Proc. 2010 International Joint Conference on Neural Networks*, 2010.

[12] A.D. Rast, X. Jin, F. Galluppi, L.A. Plana, C. Patterson, and S.B. Furber. Scalable event-driven native parallel processing: The spinnaker neuromimetic system. In *ACM International Conference on Computing Frontiers 2010*, 2010.

[13] Sen Song, Kenneth D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3:919 – 926, 2000.