Energy-efficient FPGA Spiking Neural Accelerators with Supervised and Unsupervised Spike-Timing-Dependent-Plasticity

YU LIU, SAI SOURABH YENAMACHINTALA, and PENG LI, Texas A&M University

The liquid state machine (LSM) is a model of recurrent spiking neural networks (SNNs) and provides an appealing brain-inspired computing paradigm for machine learning applications. Moreover, operated by processing information directly on spiking events, the LSM is amenable to efficient event-driven hardware implementation. However, training SNNs is in general a difficult task as synaptic weights shall be updated based on neural firing activities while achieving a learning objective. In this paper, we explore bio-plausible spike-timing-dependent-plasticity (STDP) mechanisms to train liquid state machine models with and without supervision. First, we employ a supervised STDP rule to train the output layer of the LSM while delivering good classification performance. Furthermore, a hardware-friendly unsupervised STDP rule is leveraged to train the recurrent reservoir to further boost the performance. We pursue efficient hardware implementation of FPGA LSM accelerators by performing algorithm-level optimization of the two proposed training rules and exploiting the self-organizing behaviors naturally induced by STDP.

Several recurrent spiking neural accelerators are built on a Xilinx Zync ZC-706 platform and trained for speech recognition with the TI46 speech corpus as the benchmark. Adopting the two proposed unsupervised and supervised STDP rules outperform the recognition accuracy of a competitive non-STDP baseline training algorithm by up to 3.47%.

CCS Concepts: • **Computing methodologies** → **Neural networks**; • **Computer systems organization** → **Neural networks**; • **Hardware** → **Neural systems**;

Additional Key Words and Phrases: Liquid State Machine, On-chip Training, Spike-Timing-Dependent-Plasticity, FPGA Accelerators

1 INTRODUCTION

The biological brain offers a promising source of inspiration for building novel hardware architectures and algorithms of next generation computing systems. Biological neurons perform complex interactions at large scales and conduct sophisticated tasks with impressive energy and space efficiency. Their exhibiting behaviors and properties are currently studied in significant research efforts aiming to model them with modern analogical tools.

A preliminary version of this article was presented at *2017 International Joint Conference on Neural Networks (IJCNN)* [12].
This material is based upon work supported by the National Science Foundation under Grant No. 1639995 and the Semiconductor Research Corporation (SRC) under task # 2692.001.
Authors' addresses: Y. Liu, Y. Sai, and P. Li, Department of Electrical and Computer Engineering, 400 Bizzell St, Texas A&M University, College Station, Texas, 77843, USA; Emails: {yliu129, ysaisourabh, pli}@tamu.edu .

XXXX-XXXX/2019/4-ART1 $15.00
https://doi.org/https://doi.org/10.1145/3145479

Recently, an increasing attention has been attracted to the concept of reservoir computing, which provides a bio-inspired computational model for exploiting the power of recurrent neural networks [16, 18]. The liquid state machine (LSM) is one specific model of reservoir computing which operates on spiking neurons.

As shown in Fig. 1, the LSM consists of two major parts. The reservoir, in which a number of spiking neurons are randomly wired up to resemble the recurrent topologies of cortical microcircuits, provides a complex nonlinear dynamics and maps the input into a high-dimensional response. The readout layer receives reservoir responses as inputs for final classification. In the conventional LSM model, only the synapses from the reservoir to the readout layer are trainable to relax the challenge of training the complex recurrent reservoir. The LSM is especially competent for spatiotemporal pattern classification applications such as speech recognition [10, 25, 27].

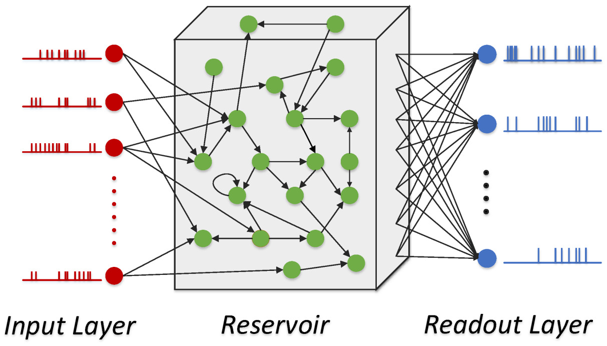

Fig. 1. A model of the liquid state machine.

Due to their power efficiency and inherent information encoding scheme, spiking neural networks (SNNs) have been targeted for dedicated silicon-based implementation on both analog and digital hardware. For instance, the Neurogrid mixed-analog-digital multi-chip system [4] realized neural elements with analog electronic circuits and transmit the axonal arbors with digital spikes, [23] developed an analog SNN for reinforcing the performance of conventional cardiac synchronization therapy devices. While analog circuits take the advantage of the inherent characteristics of silicon devices and provide low-power SNNs hardware realization, the computing accuracy is generally limited, especially for complex real-world applications such as image classification and speech recognition. On the other hand, examples of digital VLSI SNN implementations include IBM's TrueNorth chip [1] and Intel's Loihi [8]. However, both of them hold their own limitations to fully tap the computational power of spiking neural networks. The TrueNowth chip lacks integrated on-chip training capability and can only perform inference on the hardware; and no competitive on-chip training results on the real-world applications have been demonstrated by the Loihi chip by far. Generally speaking, while SNNs holding a lot of promise due to their closer resemblance to biological neurons than older generations of artificial neural networks, training spiking neural networks, especially recurrent spiking neural networks, to achieve the state-of-the-art performance remains a very difficult challenge.

To this end, the LSM is envisioned as a good trade-off between the ability in exploiting the power of recurrent spiking neural networks and engineering tractability. Recently, the unique architectural and functional properties of the LSM have been leveraged for cost-effective hardware implementations with integrated efficient on-chip spike-dependent learning mechanisms to tune the reservoir and the readout layer [13, 14, 26, 27]. However, a key limitation of the output training algorithms implemented in these works is that good performance is typically guaranteed only with full connectivity between the reservoir and readout. This leads to overall high complexity of the network and also large overhead for hardware implementation. Besides, training algorithms that applied to the LSM and SNNs in general shall update the synaptic weights only based on the local neural firing activities while achieving the end learning objectives. This natural property of the SNN imposes a significant challenge on the design of learning algorithms, as most conventional optimization methods do not satisfy it.

The above challenges motivate us to seek an alternative learning algorithm. To this end, spike-timing-dependent plasticity (STDP) [5], a well-known unsupervised learning mechanism can be considered as a good solution if combined with supervision given that it operates by locally tuning synaptic weights according to temporal spike correlations and produces interesting self-organizing behaviors. Ideas of combining supervision and STDP have been explored for precisely timed spike pattern reproduction and decision making [9, 19, 20], however, without demonstrating in real-world applications.

A preliminary version of the work in this paper has been presented in [12] that proposed the calcium-modulated supervised STDP particularly under the context of the LSM, which was only evaluated in software simulation with continuous values and STDP learning curves. In this paper, we explore STDP mechanisms to train liquid state machine models with and without supervision on a hardware LSM accelerator. First, we employ a supervised STDP rule to train the output layer of the LSM while delivering good classification performance. Furthermore, a hardware-friendly unsupervised STDP rule [13] is leveraged to train the recurrent reservoir for a further performance boost. We pursue efficient hardware implementation of FPGA LSM accelerators which allows for on-chip training and inference by performing algorithm-level optimization of the two proposed training rules and exploiting the self-organizing behaviors naturally induced by STDP. The runtime on-chip learning accuracy as well as the hardware implementation overhead of the LSM neural processors are reported in this paper.

The implemented calcium-modulated supervised STDP algorithm for the output layer targets two important objectives: delivering good learning performance and sparsifying output synapses to reduce network complexity and potentially hardware overhead. By nature, these two objectives are competing with each other as sparsifying readout synapses can easily harm learning performance. To address this challenge, a unifying two-step supervised STDP tuning approach is adopted such that both objectives can be achieved at the same time.

Towards the objective to improve the learning performance, a calcium-modulated learning algorithm based on supervised STDP is proposed, denoted as *CaL-S^{2}TDP*. In *CaL-S^{2}TDP*, for a given input class, the supervisory signal applied to the targeted output neuron and instructs it to fire at a high frequency level. For the undesired output neurons that associated to different labels with the presented input, motivated by the STDP mechanisms discovered in the brain [7], we define a depressive STDP learning rule to force them to fire at a desired low frequency level. In this way, we maximize the distance of firing frequencies from the desired neuron to undesired neurons for making classification decisions.

Moreover, the proposed *CaL-S^{2}TDP* deals with the weight saturation problem that has not been addressed in earlier supervised STDP algorithms. The weight saturation in SNNs prevents neurons from learning new information in its later training stage [6] and can result in a poor learning performance. The weight saturation problem is even worse on hardware SNNs as the synaptic weight resolution is limited. Furthermore, frequent weight updates lead to excessive memory access on the hardware neural processor and hence increase the power consumption. To solve these problems, in the proposed *CaL-S^{2}TDP* algorithm, we employ a probabilistic weight update scheme and a calcium-modulated stop-learning mechanism to slow down the weight update and hence the learning progress.

In general, a full connection between the reservoir and the readout layer with high bit resolutions are required for good classification performance. This could potentially lead to the over-fitting problem due to the high network complexity, and also brings a large silicon overhead and energy dissipation on hardware LSM accelerators. Towards the objective of sparsifying output synapses to reduce network complexity and potentially hardware overhead which tackles this issue, we propose a calcium-modulated sparsification algorithm based on supervised STDP, denoted as *CaS-S^2TDP*. The *CaS-S^2TDP* algorithm sparsifies the readout connectivity to a noticeable degree and demonstrates that it can reduce power consumption without significantly degrading the learning performance.

The key ingredient of the proposed supervised STDP readout training algorithm is that we cascade the *CaS-S^2TDP* and *CaL-S^2TDP* training and achieve the aforementioned two competing objectives simultaneously through a unifying two-step supervised STDP based readout tuning approach. Essentially, *CaS-S^2TDP* exploits the automatic competition among afferent synapses of each readout neuron induced by the STDP weight tuning mechanism [22]. This as a result produces a synaptic weight dynamics with desired sparsity while preserves the spatiotemporal structure in the input. The sparsity discovered by *CaS-S^2TDP* is carried over to the training under *CaL-S^2TDP*.

While STDP in general is amenable to hardware realization given its simplicity and locality, realizing continuous STDP in a digital architecture with cost effectiveness still poses a substantial challenge. An accurate implementation with high resolution costs large hardware overhead. However, utilizing low bit resolution by sparsely sampling the continuous weight and STDP curve could easily harm the learning performance. To address this challenge, in this work, we perform the algorithm-level optimization for the two STDP training algorithms and also leverage the self-organizing behaviors naturally induced by STDP. In the reservoir, the data-driven hardware-optimized STDP [13] is adopted, which gives a low bit-resolution hardware realization with minimum discretization errors. In the readout layer, we design the learning engine with minimized resource and power overhead by maximizing the resource sharing among different learning processes.

Several FPGA recurrent spiking neural accelerators are built on a Xilinx Zync ZC-706 platform with the ARM microprocessor on the same board serving as the host. For demonstration purpose, the neural accelerators are trained for the non-trivial speech recognition task with the TI46 [24] speech corpus benchmark. Our results indicate that the LSM neural accelerators can achieve up to 3.47% classification performance boost with two unsupervised and supervised training algorithms compared to the baseline. Besides, we also show that both unsupervised and supervised STDP algorithms can be implemented on the hardware with great efficiency.

2 HARDWARE-FRIENDLY UNSUPERVISED STDP FOR RESERVOIR TRAINING

In this section, we briefly introduce the standard STDP as a reference and then discuss the implemented hardware-friendly unsupervised STDP for reservoir tuning.

2.1 Baseline Unsupervised STDP

The nearest-neighbor STDP is an unsupervised Hebbian learning mechanism that updates the weight of a synapse based on the relative spiking timing of its pre- and postsynaptic neurons [5]. For a given synapse connected from neuron j to neuron i, the weight gets update on both pre- and postsynaptic spike events and the amount Δw relies on the temporal difference $\Delta t = t_i - t_j$ between the spike pair:

$$\Delta w^{+} = A_{+}(w) \cdot e^{-\frac{|\Delta t|}{\tau_{+}}} \; if \Delta t > 0$$
$$\Delta w^{-} = A_{-}(w) \cdot e^{-\frac{|\Delta t|}{\tau_{-}}} \; if \Delta t < 0, \tag{1}$$

where Δw^{+} and Δw^{-} denote the weight update value caused by long-term potentiation (LTP) and long-term depression (LTD), and $A_{\pm}(w)$ determines the strength of LTP/LTD, respectively. A typical STDP characteristics is plotted in Fig. 2(a).

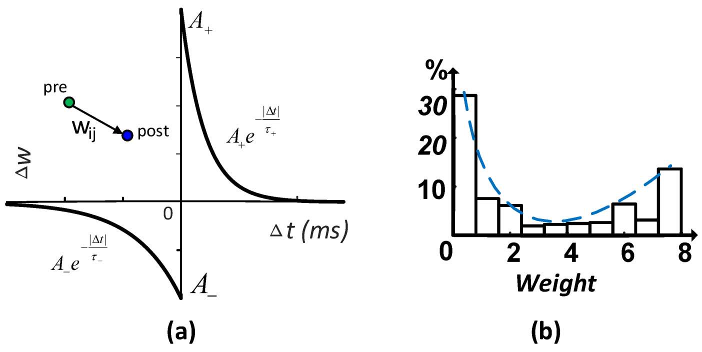

Fig. 2. (a) The standard STDP curve. (b) The equilibrium weight distribution obtained by applying STDP on a reservoir.

By nature, STDP introduces self-organizing behavior to the network by inducing competition among the afferent synapses of a neuron and can potentially lead to a sparse network topology. This is exploited by us in both the reservoir and the readout layer to build an energy-efficient neural processor, which will be introduced in Section 2.2 and Section 3.3, respectively. To give an impression on the obtained sparse structure, we apply standard STDP on the reservoir and plot the converged synaptic weight distribution in Fig. 2(b). Generally, as a common practice, only excitatory synapses are tunable while inhibitory synapses are fixed for a stabilized network dynamics. The resulting bimodal weight distribution indicates a considerable amount of zero-valued and low-valued synapses, which can be dropped off in the following training stages to save hardware energy.

2.2 Proposed Hardware-Friendly STDP for Reservoir Tuning

[11] demonstrates that tuning reservoir with unsupervised STDP can supply the readout training thus boost the learning performance [11]. Moreover, the self-organizing behavior introduces the sparsity into the reservoir and improve the hardware implementation efficiency. However, a cost-effective hardware realization of a given STDP presents a challenge. On one hand, straightforward hardware implementation in high bit resolution gives a good performance, however, at a cost of large power/area overhead. On the other hand, simply reducing the overhead by implementing the algorithm with low bit resolution leads to an immediate performance drop. To address this challenge, we adopt the hardware-optimized STDP realization with low bit resolution based on a data-driven approach [15], which is briefly summarized here.

Implementation STDP on the hardware neural processor requires both synaptic weights and the learning curve to be discretized, which introduces aggregated quantization errors. To solve this, the adopted hardware-optimized STDP algorithm discretizes the synaptic weight and the learning curve collaboratively in a data-driven approach so as to match realistic synaptic events and minimize quantization error over a large set of STDP updates. This includes 1) discretizing the continuous weights such that the equilibrium weight distribution is well represented, 2) discretizing the STDP curve to match the characteristics of the synaptic update given the spike timing difference Δt and the continuous weight change Δw.

The pseudo code of the hardware-friendly STDP algorithm is presented below. The weight w and weight change Δw with superscript d refer to the discretized values, while those with the superscript c represent the continuous ones. The synaptic weight resolution B is usually chosen to be very small for resource and power efficiency. In this way, the STDP curve is mapped to a look-up table (LUT) for weight update in the hardware. This optimization problem can be solved offline given the small design space.

ALGORITHM 1: Hardware-friendly STDP Algorithm

```
begin
    STEP 1: Profile continuous STDP:
    Run continuous STDP simulation with typical inputs, collect synaptic events:
    $\{\Delta t_k, \Delta w_k^c, w_{old,k}^c, w_{new,k}^c\}$, $k \in [1, N]$, and weight distribution
    STEP 2: Optimize weight discretization:
    Set digital reservoir synaptic weight resolution $B$
    foreach $w_k^c$ do
        $\underset{w_j^d}{\text{minimize}} \sum_k \min_{w_j^d}\{(w_k^c - w_j^d)^2\}, \ w_j^d \in [w_{min}, w_{max}], j \in [1, 2, \cdots, 2^B]$
    end
    STEP 3: Optimize STDP learning curve:
    Set digital reservoir synaptic weight resolution $B$
    foreach $\{w_{old,k}^d, \Delta t_k\}$ do
        $\underset{w_{new,k}^d}{\text{minimize}} \sum_k \min_{w_{new,k}^d} \{(w_{new,k}^c - w_{new,k}^d)^2\}, \ w_{new,k}^d \in \{w_1^d, w_2^d, \cdots, w_{2^B}^d\}$
    end
end
```

3 HARDWARE-FRIENDLY SUPERVISED STDP FOR READOUT TRAINING

In SNNs, information is encoded and processed in the form of local spikes. This enforces synaptic weights to be updated locally based on neural firing activities when training SNNs. Under this consideration, STDP, which by nature locally tunes the synaptic weight according to temporal spike correlations, can serve as a good alternative to train SNNs towards certain learning objectives. However, how to apply supervision on the by-default unsupervised STDP mechanism needs carefully study, which we present in this section.

3.1 Baseline Supervised STDP

Classification decisions made by the LSM can be inferred from the associated class label of the output neuron with the highest firing frequency. Given that, we describe the target of a supervised training algorithm on spiking neural networks as: maximizing the firing frequency of the readout neuron whose class label corresponds to the presented input sample, referred to as the "desired neuron", and at the same time minimizing the firing frequency of all other readout neurons, referred to as "undesired neurons".

Mathematically, this is to solve the following optimization problem:

$$\max_{f_j^i} \sum_{i=1}^{N} (f_{c(i)}^i(X_i, W) - \sum_{j \neq c(i)}^{C} f_j^i(X_i, W)), \text{ subject to} f_j^i \geq 0, \tag{2}$$

where N is the total number of training samples, C is the total number of input classes, and X_i is the i_{th} input sample that belongs to class $c(i)$. f_j^i is the firing frequency of the j_{th} readout neuron under the i_{th} input, and W is the readout synapse weight vector.

In Eqn. 2, for each input sample, we want to maximize the distance of firing rate between the desired neuron and undesired neurons so as to optimize the classification error over the entire training dataset. However, solving it in a mathematically exact manner is formidable.

Therefore, instead of solving Eqn.2 directly, we propose the deterministic supervised STDP algorithm, referred to as $D - S^2TDP$, which is a feasible solution exploiting the local weight update characteristics of STDP (Fig. 3(a)). The main idea of the *D-S*^{2}*TDP* is based on the observation that the standard STDP rule works by adjusting the strength of the synaptic connection between a neuron pair based on their relative firing timing. This can be leveraged to control the firing activities of the postsynaptic neuron, in our case the desired output neuron, to an expected level if a well-defined supervisory signal is given. The supervisory signal, i.e., classification teacher (CT) signal in Fig. 3(a), is an injected positive current to force the desired neuron to fire frequently and hence invoke enough weight updates. Under the mediation of the STDP, afferent synapses of the desired output neuron form a stronger connection which in turn further increases the likelihood of the postsynaptic neuron to fire in presence of its presynaptic spikes. As illustrated in Fig. 3(b), with the CT presented, the desired neuron i_1 generates more spikes in response to a presynaptic spike, resulting in further potentiation of w_{i_1}. The presence of CT also robustly bring up the learning process when the initial weights are very small.

In terms of undesired neurons, we want to prevent them from firing when unassociated input samples are presented. To achieve this, a novel depressive STDP rule is proposed (see Fig. 3(a)) to depress afferent synapses so that the chance of postsynaptic firing is reduced. As depicted in Fig. 3(c), when the undesired postsynaptic neuron i_2 fires in response to a causal spike pattern, the afferent synaptic weight w_{i_2} is decreased to discourage it to fire again.

The depression induced by the anti-causal (i.e., post-before-pre) spike pairs still applies to both desired and undesired neurons. This enables competition among plastic synapses such that a sparse structure can be learned [22].

3.2 Proposed *CaL-S*^{2}*TDP* Training Algorithm

The proposed *D-S*^{2}*TDP* effectively serves the supervised training purposes on spiking neurons. However, the deterministic weight update scheme could result in several known issues such as poor memory retention, weight saturation and large dynamic hardware power consumption. To address these problems, we optimize the supervised STDP algorithm with the proposed *CaL-S*^{2}*TDP* algorithm.

In *D-S*^{2}*TDP*, the desired output neuron maintains a high firing frequency and hence frequently update its synaptic weights. However, the number of weight levels is limited by the finite resolution representation when implemented on the hardware. As a result, the learning ends up in a way that most recent information presented to the neuron are learned better than the past information [2, 3]. This issue is known as the memory retention. Moreover, when training on the hardware, the frequent weight update results in frequent switching activities of the associated signals and logic cells as well as intensive weight memory access, which leads to high power consumption. To this

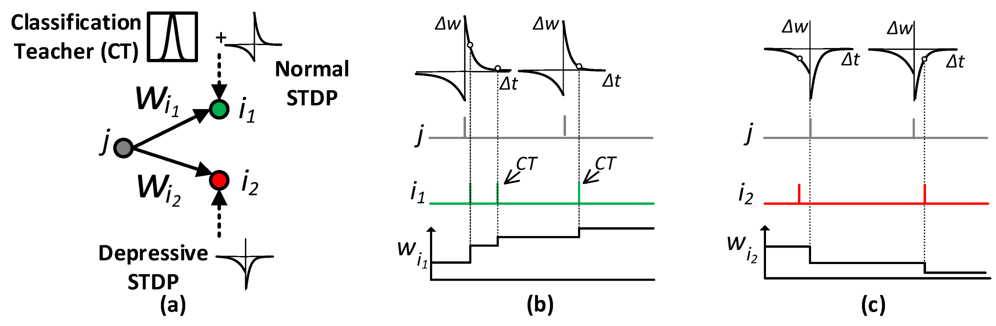

Fig. 3. (a) Proposed *D-S^{2}TDP* algorithm. The neuron i_1 is the desired neuron and i_2 is the undesired neuron. (b) and (c) Weight update under the proposed *D-S^{2}TDP* algorithm. The potentiation or depression keeps updating synaptic weights when a valid spike pair is presented.Besides, By applying CT, the spike event of the desired neuron happens periodically.

end, we adopt the probabilistic weight update scheme in [11] to slow down the learning process for better learning performance and hardware power efficiency.

Moreover, without any stop-learning mechanism, readout synapses are continuously tuned by the supervised STDP with the on-going reservoir responses and the synaptic weights are pushed to a bimodal distribution (Fig. 2) by STDP by nature. Ultimately, readout neurons will be unable to respond to any new stimuli since most of their afferent synapses are saturated at the maximum/minimum weight values.

To solve the weight saturation problem, we disable the potentiation of a synapse when its postsynaptic neuron is very active. Similarly, the depression stops when the postsynaptic neuron is already silent. Inspired by [6], in our work, the internal calcium concentration of a neuron is used to indicate its average firing level over a long time interval and to manage the activation of the learning. The calcium concentration $c(t)$ is defined as:

$$\frac{dc(t)}{dt} = -\frac{c(t)}{\tau_c} + \sum_i \delta(t - t_i), \tag{3}$$

where τ_c is the time constant and t_i is the time when the postsynaptic neuron fires. The internal calcium concentration level of the neuron increases with its firing frequency.

Given the above considerations, we integrate the calcium-modulated weight update in the supervised STDP readout training algorithm. First, a calcium threshold c_θ is defined to separate active neurons from inactive ones. Then, an activation margin δ is set. Synapse potentiation is allowed when $c < c_\theta + \delta$ and depression is allowed when $c > c_\theta - \delta$. Following the principle of Hebbian learning, we also define the lower bound of the potentiation activation range and the upper bound of c for depression. Combining the stop-learning mechanism and probabilistic weight updates, the *CaL-S^{2}TDP* algorithm is defined as:

$$\begin{aligned} w &\leftarrow w + d \quad w/prob. \propto |\Delta w^+|, \;\; if \, \Delta t > 0 \;\&\& c_\theta < c < c_\theta + \delta \\ w &\leftarrow w - d \quad w/prob. \propto |\Delta w^-|, \;\; if \, \Delta t < 0 \;\&\& c_\theta > c > c_\theta - \delta, \end{aligned} \tag{4}$$

where $\Delta w^+/\Delta w^-$ are the weight adjustments determined by the STDP rule. They further determine the probabilities of a weight update for LTP and LTD, respectively.

For the undesired neuron, since the depressive STDP is employed for both causal and anti-causal spike pair patterns, only the second equation in Eqn. 4 applies. The weight update in *CaL-S^{2}TDP* algorithm is illustrated in Fig. 4(c) and (d), where, unlike *D-S^{2}TDP* as shown in Fig. 3(b) and (c), no weight update is allowed if the calcium level is too low or too high.

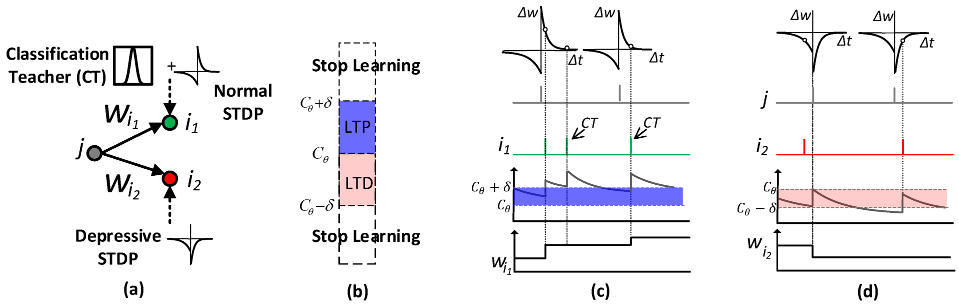

Fig. 4. (a) Proposed *CaL-S^{2}TDP* training algorithm. The neuron i_1 is the desired neuron and i_2 is the undesired neuron. (b) The calcium-modulated activation range. (c) and (d) Weight update of desired and undesired neurons. The potentiation or depression only happens when the postsynaptic calcium level c is in the activation range.

3.3 Proposed *CaS-S^{2}TDP* Sparsification Algorithm

In an LSM, synapses from the reservoir to the readout layer are fully connected and their weight resolutions are usually high to achieve good learning results. This could result in two problems: overfitting due to the high model complexity, and large hardware implementation overhead. However, randomly dropout readout synapses can significantly degrade the learning performance. Therefore, we want to design an algorithm to smartly prune readout synapses. The major difference between a sparsification algorithm from a classification algorithm is that the objective of the sparsification algorithm is to allow sufficient competition among synapses rather than to learn certain input patterns.

We realize that the STDP algorithm by nature mediates afferent synapses of a neuron to characterize competitions among them. Some synapses are strengthened while others are weakened [22]. As a result, it leads to a bimodal weight distribution (see Fig. 2 as an example) out of which many zero-valued or small-valued synapses can be pruned out. Therefore, the tuning mechanism of STDP can be leveraged in our work to develop a supervised readout sparsification algorithm. Moreover, in order to embed the sparsification into real-world classification tasks, the designed algorithm should take the spatiotemporal structures in the training samples into consideration such that the discovered sparse patterns fit well with the features represented by the reservoir responses. Working towards this target, we recognize that it is only necessary to instruct each readout neuron to learn the sparse structure of the input subset of its associated class. This leads to the maximum sparsity and the information from other classes will not be mistakenly learned through the sparsification process.

Given above considerations, we proposed the *CaS-S^{2}TDP* algorithm for readout sparsification learning. The external supervised sparsification teacher (ST) signal (Fig. 5(a)) is introduced in

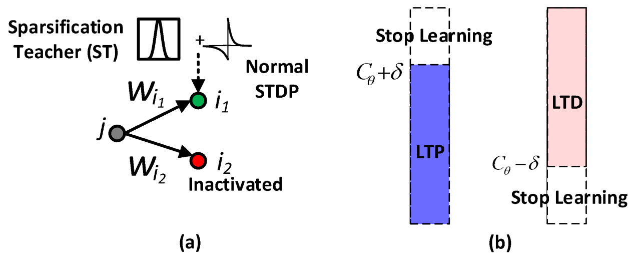

Fig. 5. (a) The *CaS-S^{2}TDP* sparsification algorithm. The activity level of the selected readout neuron i_1 is boosted by the sparsity teacher (ST). (b) Stop learning for readout synapse sparsification.

CaS-S^{2}TDP to ensures that only the afferent synapses of the desired output neuron are under sparsification learning at a time and also bring up initial firing events of each readout neuron.

To maintain good learning performance, the stop-learning mechanism is also included in the *CaS-S^{2}TDP* algorithm as shown in Fig. 5(b). However, compared to *CaL-S^{2}TDP*, the activation range of calcium concentration is more relaxed for both LTP and LTD to maximize sparsity at the same time avoid undesirable bias in calcium regulation. In conclusion, the resulting *CaS-S^{2}TDP* sparsification algorithm is summarized as:

$$\begin{aligned} w &\leftarrow w + d \ w/prob. \propto |\Delta w^+|, \ if \Delta t > 0 \ \&\& c < c_\theta + \delta \\ w &\leftarrow w - d \ w/prob. \propto |\Delta w^-|, \ if \Delta t < 0 \ \&\& c_\theta - \delta < c. \end{aligned} \tag{5}$$

3.4 Two-step Hardware-Friendly Supervised Readout Training

Combining *CaS-S^{2}TDP* with *CaL-S^{2}TDP* algorithms, we propose a unifying supervised STDP readout training approach executed in two steps seamlessly. First, *CaS-S^{2}TDP* is applied to the output layer. At the end of the sparsification, the zero-weight synapses are removed and the remaining synapses are trained by the *CaL-S^{2}TDP*.

In the proposed readout training algorithm, *CaS-S^{2}TDP* learns to capture the spatiotemporal structures of the input spikes through the self-organizing behavior of STDP. Therefore, unlike random synapse dropout, the discovered sparsity from the sparsification step can be passed to the classification training step thus degrade the learning performance as little as possible.

Realizing *CaS-S^{2}TDP* and *CaL-S^{2}TDP* on hardware entails efficient implementation of the STDP learning curve and the stochastic weight update scheme. Inspired by the realization of the unsupervised STDP algorithm in the reservoir, for the proposed supervised STDP algorithms, the weight update probability calculation is implemented by a lookup table whose entry values are carefully chosen offline according to the associated learning curve. In our design, there is a lookup table for LTD and LTP process respectively. Moreover, to minimize the resource and power overhead of the supervised STDP implementation, we use the same learning engine for both sparsification and classification training in each readout neuron. This involves resource sharing and execution time interleaving of *CaS-S^{2}TDP* and *CaL-S^{2}TDP*, which will be introduced in more detail in Section 5.2.

4 HARDWARE IMPLEMENTATION ARCHITECTURE

In this section, we describe the overall architecture of the LSM processor and the realization of digital spiking neurons.

4.1 Overall LSM Processor Architecture

Fig. 6 depicts the overall architecture of the LSM neural processor. The reservoir and the readout layer are implemented by a reservoir unit (RU) and a training unit (TU), respectively. Each reservoir neuron is implemented with a reservoir element (RE) and the output neuron is implemented with the output element (OE). The synaptic connectivity from the external input to the RU is specified by a pre-defined crossbar interface. The spiking responses generated from REs are registered and sent to all OEs. Meanwhile, these spikes are also fed back to some reservoir neurons through reservoir synapses, the connectivity of which is specified by another pre-defined crossbar. Neurons at the same layer work in parallel to fully exploit the inherent abundant parallelism of the spike-based learning algorithm.

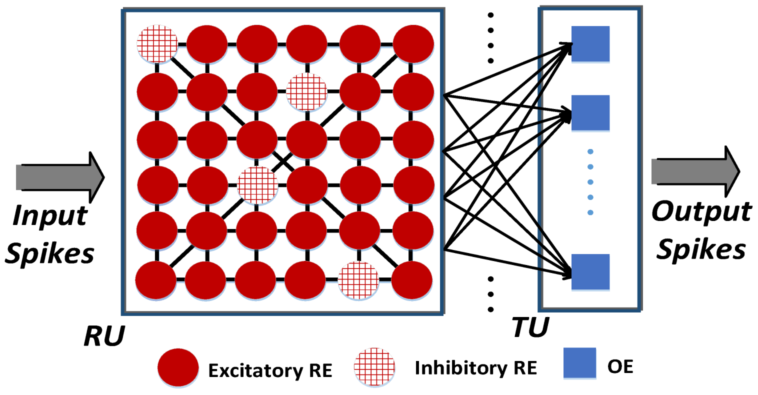

Fig. 6. Overall architecture of an exemplary LSM neural processor.

The RU and the TU are trained in two stages in the LSM neural processor. First, RU is trained until the synaptic weight distribution converges. Then, the readout training stage starts, which can be further divided into the sparsification training phase and classification training phase, in which the TU is trained by the *CaS-S^{2}TDP* and *CaL-S^{2}TDP* algorithm, respectively. At the end of the readout sparsification training phases, the zero-values readout synapses will be dropped out as the network continues to the classification training phase. These dropped out synapses are not used for inference as well. During the entire readout training stage, RU is activated to provide spike inputs to TU while maintaining its synaptic weights.

4.2 Implementation of Digital Spiking Neurons

The proposed LSM neural accelerator operates through a series of computational steps and requires a large number of storing elements inside each neuron. As shown in Fig. 7, a digital neuron module (RE or OE) contains three sub-modules that perform different functions at each biological time step: input spike processing, neuron update and spike generating, and synaptic weight learning. These three operations span across several well-defined computational steps controlled by the corresponding states associated with the global FSM at each layer. The unique architectural and functional properties of the proposed LSM neural processor naturally lead to well-defined boundaries between these sub-modules in terms of execution and storage. At each emulation time step, first, the synaptic input processing module computes the second-order synaptic spike responses with the arrival of spike inputs. Then, the spike generation module updates the membrane voltage with the synaptic responses and generates spikes based on the widely used leaky integrate-and-fire (LIF) model. At last, the learning module tunes the afferent presynaptic weights of the associated neuron.

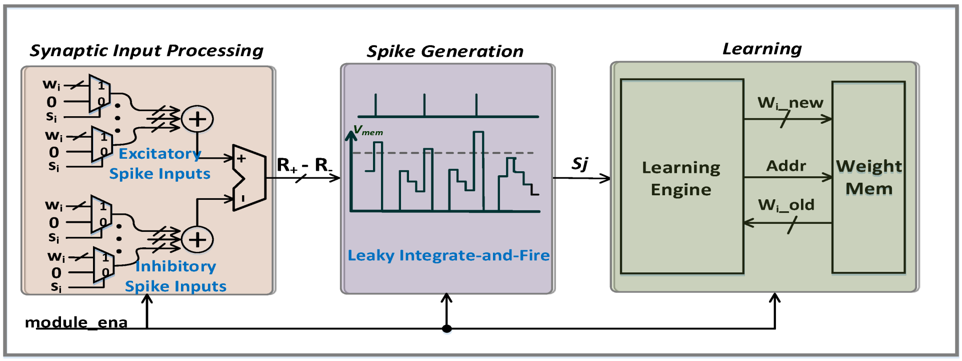

Fig. 7. Hardware implementation of a single digital neuron element.

The major difference between the RE and the OE lies in the learning module, specifically the implemented learning functions, arithmetic resolutions of digital synapses, and the realization of the weight memory. A block memory (BRAM on FPGAs) inside each OE is used to store all its presynaptic weights. REs, on the other side, make use of flip flops (FFs) to store the weight because of the lower synaptic bit resolution and fewer input synapses per neuron.

5 CIRCUIT LEVEL OPTIMIZATION

In this section, we discuss the cost-effective hardware implementation of the presented training mechanism in the RE and OE, respectively.

5.1 Implementation of Unsupervised STDP

The learning engine in the LE implements the proposed hardware-friendly unsupervised STDP reservoir tuning mechanism [15], as depicted in Fig. 8(a). In the implementation, shift registers (SRs) are used to calculate the Δt in Eqn. 1 so that a heavily loaded and frequently switching global clock counter can be avoided to save power. Assuming the number of afferent synapses of a postsynaptic neuron is m, the presynaptic shift registers SR_1 to SR_m tracks the associated presynaptic spikes and the postsynaptic shift register (i.e., SR_0) is used for tracking firing events of the neuron itself in which the learning module is instantiated. The depths of pre- and postsynaptic shift registers is decided by time windows for LTP and LTD, respectively.

At each biological time step, the learning module checks each shift register in a serial manner and updates the synapse weight if a valid spike pair presents. The time difference of a spike pair is calculated by comparing the location of “spikes" in the shift register. When a neurons fires, the MSB of its affiliated shift register is set to ‘1’ and the register shifts one bit to the right at every biological step of the network. All shift registers in the reservoir neurons are driven by the global clock for spike synchronization. By examining the relative position of “spikes" in shift registers, the temporal difference Δt between pre- and postsynaptic spikes can be easily inferred. As the example in Fig. 8(b) explains, for the considered spike pair, $\Delta t = t_{post} - t_{pre} = t_j - t_i = 2$. Note that the potential weight update only happens when there is(are) ‘1’(s) at the MSB of the shift register(s), which indicates the firing of the pre- or postsynaptic neuron at the current biological time step.

5.2 Implementation of Supervised STDP

When implementing the supervised STDP learning mechanism in the OE, we make the following two observations. First, $CaS\text{-}S^2TDP$ and $CaL\text{-}S^2TDP$ share the principles in the weight update scheme

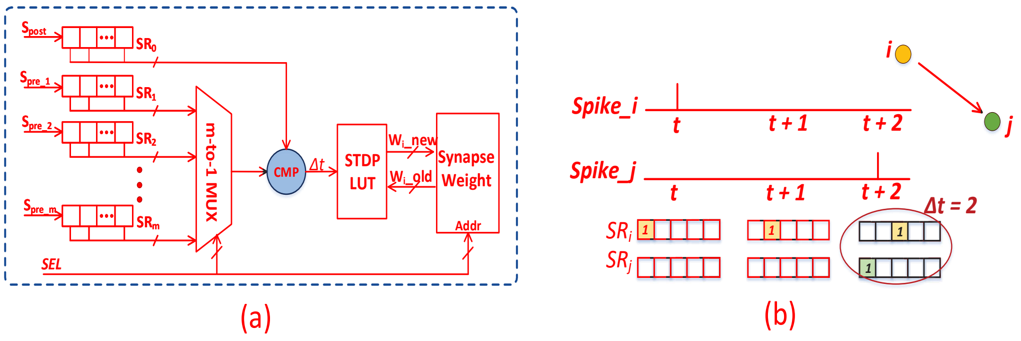

Fig. 8. (a) The design of the learning engine in LEs that implements the hardware-friendly unsupervised STDP reservoir tuning mechanism. (b) An illustration of how time difference Δt is computed in the hardware learning engine.

including the basics of STDP learning mechanism, probabilistic weight update, and the calcium-modulated stop-learning rule. Second, in the readout training stage, the sparsification training under *CaS-S*^{2}*TDP* and classification training under *CaL-S*^{2}*TDP* are executed in two phases in order without overlap. This gives us an opportunity to explore the resource sharing of logic cells and memories when implementing these two algorithms to optimize the resource utilization and power efficiency. As shown in Fig. 9, the entire data path, including arithmetic logic cells and STDP learning lookup tables (LUTs), are shared by both algorithms. Moreover, in *CaL-S*^{2}*TDP* implementation, the "potentiation" in the depressive STDP rule for undesired neurons is implemented by the same LTP LUT as the regular STDP LTP curve for calculating update probability. To realize the depression update, instead, we inverse weight update value from $+\Delta W$ to $-\Delta W$ when $\Delta t > 0$, which is controlled by the CT as shown in Fig. 9. As such, we maximize the resource reuse to build an overhead and energy efficient readout learning engine.

In the learning engine in OEs, first, we follow the implementation in the RE that computes the spike timing differences using shift registers. As shown in Fig. 9, SR$_0$ is the postsynaptic shift register and SR$_1$ to SR$_\text{m}$ are the presynaptic shift registers. In OEs, the value of m is generally much larger than that in the LE due to the full connectivity of the readout synapses.

After the spike timing difference Δt is computed, first, its signed bit is examined to determine whether this is an LTP or LTD update. LTP and LTD lookup tables store the weight update probability which is related to the time difference. In general, a smaller $|\Delta t|$ indicates a stronger relation between the pre- and postsynaptic neurons thus leads to a higher weight update probability according to the STDP tuning mechanism. The entries of both look-up tables are optimized to get good learning performance. At each biological time step, at most one LUT is enabled. The LUTs are implemented with the distributed RAM on the FPGA with zero read latency. The weight update probability output from the LUT is then compared it with the output from the random number generator (i.e., RNG in Fig 9), which is implemented by a linear-feedback shift register that generates a different pseudo-random number at each biological time step. If the generated random number is smaller than the probability threshold, and at the same time the calcium concentration is in the activation range, then the corresponding synaptic weight is updated. Similar to the RE, the calculation of Δt and Δw in OE are executed in serial in the order of synapse index in each neuron.

Note that during the readout sparsification phase, only the afferent synapses of the desired readout neuron are enabled for weight update. Therefore, the ST signal in 9 serves as an enable

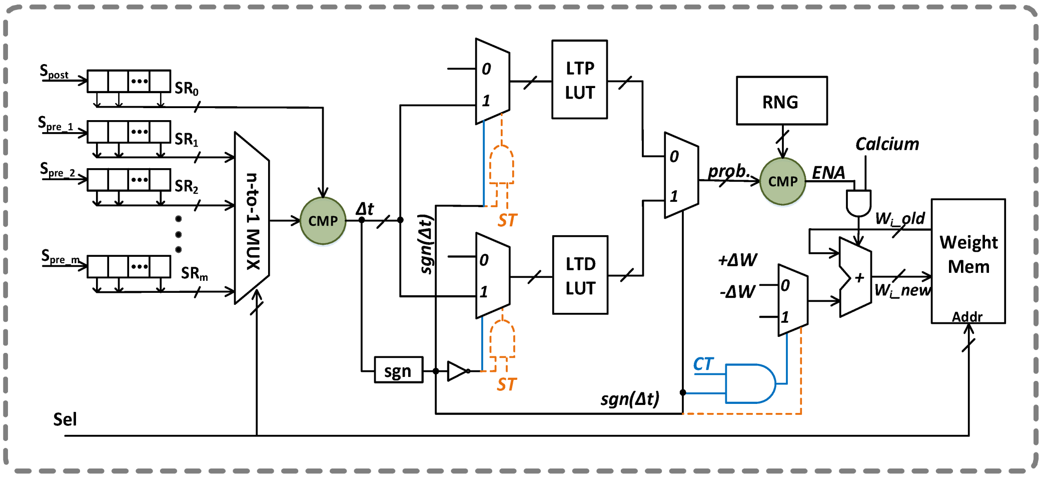

Fig. 9. Implementation of the proposed *CaS-S²TDP* and *CaL-S²TDP* algorithm. The blue path are the control path specified to *CaL-S²TDP* and the orange path are specified to *CaS-S²TDP*. The black paths represent the data and control path shared by two algorithms.

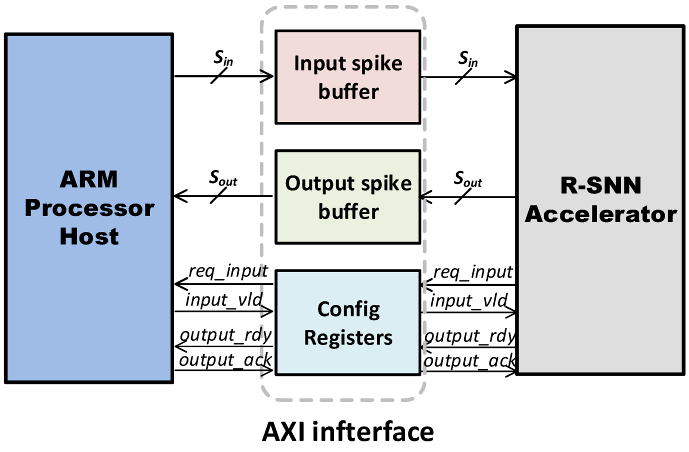

Fig. 10. The illustration of the recurrent spiking neural computing system.

signal for the STDP LUTs and the following data path. If ST equals to 0, the entire weight update logic stays inactivated.

6 EXPERIMENTAL SETTINGS AND BENCHMARKS

6.1 Recurrent Spiking Neural Accelerator Setup

In this work, several LSM neuromorphic processors are built on a Xilinx Zync ZC-706 platform as FPGA accelerators. The onboard ARM Cortex-A9 MPCore microprocessor serves as the host for the FPGA neural accelerator to provide input data and to receive the output spikes and analyze the classification performance. Fig. 10 shows the recurrent spiking neural processing system.

The LSM neural accelerator communicates with the host through a high-speed 32-bit AMBA AXI interface in a hand-shaking manner. When the host receives a request (*req_input* in Fig. 10) for a new input from the LSM accelerator, it writes the input pattern of the current biological time step to the input spike buffer located inside the interface. The depth of the input buffer is 1 and the width equals the number of spike channels of the input data. The original input files are stored in an SD card which can only be directly accessed by the host. After the input spike write is done, the host asserts an input valid signal (*input_vld* in Fig. 10) in the configuration registers (config registers in Fig. 10). This bit will be seen by the LSM accelerator and it then takes the spikes from the input buffer and starts processing. The neural accelerator is also responsible for cleaning the *input_vld* bit after reading input spikes. Before the *input_vld* signal is deasserted by the LSM accelerator, the host is blocked and would not process any other function.

During the inference stage, the host takes the output spikes generated from the LSM neural accelerator to analyze the classification accuracy. After the LSM neural processor finishes processing the current input, it asserts the *output_ready* signal. The host keeps pooling the configuration registers for this signal. When the host sees the signal asserted, it takes the spikes out from the output spike buffer and updates the spike counts of each output neurons accordingly. At the end of each input sample, the host interprets the classification decision by selecting the corresponding class label of the output spiking neuron that fires most during the presence of the current input sample. This classification decision is then compared with the ground truth label to see if it is correct. At the end of the inference stage, the host will report the overall classification accuracy as the performance of the LSM neural processor.

6.2 Training Setup and Benchmarks

In the LSM neural accelerator implemented in this work, there are 135 reservoir neurons set up on a 3D grid using the approach described in [27]. 80% of the reservoir neurons are excitatory and the rest are inhibitory. The number of readout neurons is decided by the number of classes to be classified in the benchmark, which is 26 in our case.

In the reservoir layer of the proposed recurrent spiking neural processor, we adopt the optimized hardware-friendly unsupervised STDP training from [13]. To minimize hardware implementation cost, the reservoir synaptic weights is set to 2 and weight changes are only executed when $|\Delta t| \leq 3$. Table. 1 shows the lookup table that is implemented in the LSM neural accelerator.

Table 1. Optimized weight discretization of unsupervised STDP

	$w_1^d = 0$	$w_2^d = 2$	$w_3^d = 6$	$w_4^d = 8$
$\Delta t = -3$	w_1^d	w_2^d	w_3^d	w_4^d
$\Delta t = -2$	w_1^d	w_1^d	w_2^d	w_3^d
$\Delta t = -1$	w_1^d	w_1^d	w_1^d	w_2^d
$\Delta t = 0$	w_1^d	w_2^d	w_3^d	w_4^d
$\Delta t = 1$	w_3^d	w_4^d	w_4^d	w_4^d
$\Delta t = 2$	w_2^d	w_3^d	w_4^d	w_4^d
$\Delta t = 3$	w_1^d	w_2^d	w_3^d	w_4^d

For the supervised STDP readout training approach, the parameters of the algorithms are selected by exploring the design space to a certain level and we present the chosen values of the key parameters in Table 2. To optimize the hardware overhead and at the same time guarantee a good learning performance, the readout synaptic weight is set to 10-bit signed integers for all algorithms that are studied in this paper. The initial weights are random values between W_{min} and W_{max}. The depth of both LTD and LTP LUT are set to 16 and the LUTs are tuned offline in the software simulator such that a good classification performance can be achieved.

Table 2. Parameter settings of the proposed supervised STDP algorithms.

Parameter	**Value**
A_+	3.0
A_-	1.5
τ_+	4.0
τ_-	8.0
ΔW	1
c_θ	5.0
δ	3.0
τ_c	64.0

The adopted benchmark is a subset of the TI46 speech corpus [24], which contains utterances of English letters from "A" to "Z". There are 260 samples in this benchmark, ten for each letter, recorded from a single speaker. The time domain speech signals are first preprocessed by Lyon's passive ear model [17] and then encoded into 78 spike trains using the BSA algorithm [21].

During the readout sparsification phase, the *CaS-S*^{2}*TDP* is iterated for a sufficient number until the distribution of the readout synaptic weight reaches a steady state. Based on our observation, the iteration times is set to 20, which is same as the number of iterations of the unsupervised STDP reservoir training. This will lead to 25% readout synapses to be sparsified. Then, the readout layer is trained by the proposed *CaL-S*^{2}*TDP* algorithm for another 250 iterations, during which the zero-weight output synapses will not be considered for weight update. A 5-fold cross-validation scheme is adopted when evaluating the recognition performance.

7 EXPERIMENTAL RESULTS

With the experimental settings introduced in Section 6, in this section, we report the learning performance and hardware overhead of the LSM neural accelerators with the proposed supervised and unsupervised STDP training algorithms.

7.1 Classification Performance of the Recurrent Spiking Neural Accelerator

Given the considered design space, the on-chip learning performances of several recurrent spiking neural processors for the speech recognition task with the TI46 corpus benchmark are reported in Table 3. In the table, we also show the performance boost of each training mechanism compared to the baseline design. The neural accelerators are implemented with different reservoir and readout training mechanisms as described in the table. The "X+Y" by default means applying X training mechanism to the reservoir layer and Y to the output layer of the corresponding LSM neural accelerator. The "baseline" output training algorithm is a competitive non-STDP supervised spike-dependent training algorithm proposed in [27]. In the fixed reservoir, synapses weights are not changeable and are set to 1 for excitatory synapses and −1 for inhibitory ones. The "Unsupv STDP" represents the proposed hardware-friendly unsupervised STDP reservoir training algorithm.

From the results, it is evident that both unsupervised STDP reservoir training and supervised STDP readout training algorithm can noticeably improve the classification accuracy of the LSM neural accelerator. By simply training the reservoir with the proposed hardware-friendly unsupervised STDP algorithm, we can get a performance boost of 1.93% on top of the baseline design.

Table 3. Performances of LSM neural accelerators with different training mechanisms.

	Fixed + Baseline	**Unsupv STDP + Baseline**	**Fixed +** *CaL-S^{2}TDP*	**Unsupv STDP +** *CaL-S^{2}TDP*	**Fixed +** *CaL-S^{2}TDP* **&** *CaS-S^{2}TDP*	**Unsupv STDP +** *CaL-S^{2}TDP* **&** *CaS-S^{2}TDP*
Classification Accuracy	91.53% (-)	93.46% (+1.93%)	94.23% (+2.70%)	95.00% (+3.47%)	91.92% (+0.39%)	93.84% (+2.31%)

When applying only the *CaL-S^{2}TDP* on the readout layer, the performance boost is up to 2.7%. And when we combine the STDP-based reservoir training and the readout training, we can get a major performance improvement of 3.47% on the final classification. The table also shows that with a sparsified readout connection brought by *CaS-S^{2}TDP*, the LSM neural processor can still deliver a decent learning performance which is higher than the baseline. This outperforms the LSM neural processors with randomly dropped readout synapses, in which an apparent performance degradation is observed according to the results reported in [12].

7.2 Hardware Overhead and Training Efficiency of the Recurrent Spiking Neural Accelerator

In this section, we compare the overhead of implementing different training mechanisms on the LSM neural accelerators in terms of resource utilization and dynamic power consumption. Table 4 shows the hardware resource utilizations of LSM neural processors implemented with different learning mechanisms in terms of slice flip flops (FFs) and slice LUTs as well as their percentages of usage with respect to the available resources on the targeted FPGA board. Here we only consider the resource usage of the LSM neural processor accelerator itself and the overhead of the AXI interface is not included because the interface only takes a small portion of the design and is the same among different LSM neural processors. Similarly, in Table 5, we report the dynamic training power consumption of different spiking neural accelerators which is estimated by the Xilinx Power Analyzer given the activity-based simulation results. The power results are estimated under the 100MHz clock frequency, which is consistent with the working clock frequency of the physical hardware accelerator.

Table 3, Table 4 and Table 5 in together show the trade-off between the learning accuracy and the hardware implementation overhead on the recurrent spiking accelerator of different training algorithms. From Table 4 and Table 5, we can tell that implementing the supervised STDP readout training required an extra overhead for both on-chip resources and power. The extra overhead is mainly due the cost of computing the spike timing difference Δt of pre- and postsynaptic neurons for all readout synapses. In order to achieve a decent classification performance, the time windows and correspondingly depths of shift registers reserved in proposed supervised STDP algorithms, *CaS-S^{2}TDP* and *CaL-S^{2}TDP*, are set to 12 for both LTP and LTD. This is much larger than that in the reservoir for the unsupervised STDP which is set to 3. Moreover, a full connectivity between the reservoir and the readout layer required a large number of flip flops to be utilized for implementing supervised STDP algorithms, which contributes majorly to the extra resource and dynamic power overhead. However, considering that the extra overhead of implementing unsupervised and supervised training mechanism on the LSM neural accelerator is relatively small compared to its learning accuracy boost over the baseline, and that the power and resource utilization is overall low compared to the training cost on the software simulator, the training efficiency of the hardware LSM neural accelerator is still noteworthy.

The results from Table 4 and Table 5 also shows that the proposed *CaS-S^{2}TDP* reduces the power consumption of the readout classification training stage compared to the case when only the *CaL-S^{2}TDP* is applied. Besides, the additional overhead to implement *CaS-S^{2}TDP* is very small. This indicates that by sharing the resources in the learning engine in readout neurons, we can efficiently implement the supervised STDP readout training for both sparsification and classification at the same time.

Table 4. Hardware resource utilization of LSM neural accelerators with different training mechanisms.

	Fixed + Baseline	**Unsupv STDP + Baseline**	**UnsupV STDP +** *CaL-S^{2}TDP*	**Unsupervised +** *CaS-S^{2}TDP* **&** *CaL-S^{2}TDP*
FFs	12694	12717	19841	19844
LUTs	43975	45785	57581	57788
FFs Utilization	2.90%	2.91%	4.54%	4.54%
LUTs Utilization	20.18%	20.95%	26.34%	26.43%

Table 5. Classification training power of different algorithms on LSM neural accelerators.

	Fixed + Baseline	**Unsupv STDP + Baseline**	**Unsupv STDP+** *CaL-S^{2}TDP*	**Unsupv STDP+** *CaS-S^{2}TDP* **&** *CaL-S^{2}TDP*
Training for Classification Power (mW)	161	195	237	229

8 CONCLUSION

In this paper, we explore bio-plausible STDP training mechanisms with and without supervision on LSM FPGA recurrent spiking neural accelerators. A novel two-step supervised STDP approach is implemented to train the output layer of the LSM for both classification performance and synapse sparsification, and a hardware-friendly unsupervised STDP is used to train the reservoir. The hardware efficiency of the LSM neural accelerators is optimized by performing algorithm-level optimization of the two training algorithms and exploiting the self-organizing behaviors of the STDP. Using the speech recognition task as a demonstrating application, we measure the classification performance of the proposed recurrent spiking neural accelerator built on the Xilinx Zync ZC-706 FPGA. The results demonstrate that the proposed unsupervised and supervised STDP training algorithms can work together to greatly improve the accuracy of the LSM with excellent hardware efficiency.

REFERENCES

[1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. 2015. TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 10 (2015), 1537–1557.

[2] Daniel J Amit and Stefano Fusi. 1992. Constraints on learning in dynamic synapses. *Network: Computation in Neural Systems* 3, 4 (1992), 443–464.

[3] Daniel J Amit and Stefano Fusi. 1994. Learning in neural networks with material synapses. *Neural Computation* 6, 5 (1994), 957–982.

[4] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V Arthur, Paul A Merolla, and Kwabena Boahen. 2014. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 5 (2014), 699–716.

[5] Guo-qiang Bi and Mu-ming Poo. 2001. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual review of neuroscience* 24, 1 (2001), 139–166.

[6] Joseph M Brader, Walter Senn, and Stefano Fusi. 2007. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural computation* 19, 11 (2007), 2881–2912.

[7] Natalia Caporale and Yang Dan. 2008. Spike timing-dependent plasticity: a Hebbian learning rule. *Annu. Rev. Neurosci.* 31 (2008), 25–46.

[8] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.

[9] Jan-Moritz P Franosch, Sebastian Urban, and J Leo van Hemmen. 2013. Supervised spike-timing-dependent plasticity: A spatiotemporal neuronal learning rule for function approximation and decisions. *Neural computation* 25, 12 (2013), 3113–3130.

[10] Arfan Ghani, T Martin McGinnity, Liam P Maguire, and Jim Harkin. 2008. Neuro-inspired speech recognition with recurrent spiking neurons. In *Artificial Neural Networks-ICANN 2008*. Springer, 513–522.

[11] Yingyezhe Jin and Peng Li. 2016. AP-STDP: A novel self-organizing mechanism for efficient reservoir computing. In *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 1158–1165.

[12] Yingyezhe Jin and Peng Li. 2017. Calcium-modulated supervised spike-timing-dependent plasticity for readout training and sparsification of the liquid state machine. In *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2007–2014.

[13] Yingyezhe Jin, Yu Liu, and Peng Li. 2016. SSO-LSM: A Sparse and Self-Organizing architecture for Liquid State Machine based neural processors. In *Nanoscale Architectures (NANOARCH), 2016 IEEE/ACM International Symposium on*. IEEE, 55–60.

[14] Yu Liu, Yingyezhe Jin, and Peng Li. 2017. Exploring sparsity of firing activities and clock gating for energy-efficient recurrent spiking neural processors. In *Low Power Electronics and Design (ISLPED, 2017 IEEE/ACM International Symposium on*. IEEE, 1–6.

[15] Yu Liu, Yingyezhe Jin, and Peng Li. 2018. Online Adaptation and Energy Minimization for Hardware Recurrent Spiking Neural Networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 14, 1 (2018), 11.

[16] Mantas LukošEvičIus and Herbert Jaeger. 2009. Reservoir computing approaches to recurrent neural network training. *Computer Science Review* 3, 3 (2009), 127–149.

[17] Richard F Lyon. 1982. A computational model of filtering, detection, and compression in the cochlea. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, Vol. 7. IEEE, 1282–1285.

[18] Wolfgang Maass, Thomas Natschläger, and Henry Markram. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation* 14, 11 (2002), 2531–2560.

[19] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. 2006. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural computation* 18, 6 (2006), 1318–1348.

[20] Filip Ponulak and Andrzej Kasinski. 2010. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Computation* 22, 2 (2010), 467–510.

[21] Benjamin Schrauwen and Jan Van Campenhout. 2003. BSA, a fast and accurate spike train encoding scheme. In *Proceedings of the International Joint Conference on Neural Networks*, Vol. 4. IEEE Piscataway, NJ, 2825–2830.

[22] Sen Song, Kenneth D Miller, and Larry F Abbott. 2000. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience* 3, 9 (2000), 919.

[23] Qing Sun, François Schwartz, Jacques Michel, Yannick Herve, and Renzo Dal Molin. 2011. Implementation study of an analog spiking neural network for assisting cardiac delay prediction in a cardiac resynchronization therapy device. *IEEE transactions on neural networks* 22, 6 (2011), 858–869.

[24] TI46. [n. d.]. The TI46 Speech Corpus. http://catalog.ldc.upenn.edu/LDC93S9

[25] David Verstraeten, Benjamin Schrauwen, Dirk Stroobandt, and Jan Van Campenhout. 2005. Isolated word recognition with the liquid state machine: a case study. *Inform. Process. Lett.* 95, 6 (2005), 521–528.

[26] Qian Wang, Youjie Li, and Peng Li. 2016. Liquid state machine based pattern recognition on FPGA with firing-activity dependent power gating and approximate computing. In *Internatioal Symposium of Circuits and Systems (ISCAS), 2016 IEEE.* IEEE, 361–364.

[27] Yong Zhang, Peng Li, Yingyezhe Jin, and Yoonsuck Choe. 2015. A Digital Liquid State Machine With Biologically Inspired Learning and Its Application to Speech Recognition. *IEEE transactions on neural networks and learning systems* 26, 11 (2015), 2635–2649.