Online Adaptation and Energy Minimization for Hardware Recurrent Spiking Neural Networks

YU LIU,* YINGYEZHE JIN,* and PENG LI, Texas A&M University

The Liquid State Machine (LSM) is a promising model of recurrent spiking neural networks that provides an appealing brain-inspired computing paradigm for machine-learning applications such as pattern recognition. Moreover, processing information directly on spiking events makes the LSM well suited for cost- and energy-efficient hardware implementation. In this article, we systematically present three techniques for optimizing energy efficiency while maintaining good performance of the proposed LSM neural processors from both an algorithmic and hardware implementation point of view. First, to realize adaptive LSM neural processors, thus boost learning performance, we propose a hardware-friendly Spike-Timing Dependent Plastic (STDP) mechanism for on-chip tuning. Then, the LSM processor incorporates a novel runtime correlation-based neuron gating scheme to minimize the power dissipated by reservoir neurons. Furthermore, an activity-dependent clock gating approach is presented to address the energy inefficiency due to the memory-intensive nature of the proposed neural processors.

Using two different real-world tasks of speech and image recognition to benchmark, we demonstrate that the proposed architecture boosts the average learning performance by up to 2.0% while reducing energy dissipation by up to 29% compared to a baseline LSM with little extra hardware overhead on a Xilinx Virtex-6 FPGA.

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

Additional Key Words and Phrases: Liquid state machine, online adaptation, spike-timing dependent plasticity, energy efficiency

ACM Reference format:
Yu Liu, Yingyezhe Jin, and Peng Li. 2018. Online Adaptation and Energy Minimization for Hardware Recurrent Spiking Neural Networks. *J. Emerg. Technol. Comput. Syst.* 14, 1, Article 11 (January 2018), 21 pages.
https://doi.org/10.1145/3145479

1 INTRODUCTION

The human brain has the ability to perceive, memorize, and respond to the outside world. It elegantly performs complex tasks such as describing the features of an image, understanding sophisticated sentences, adapting to the changing environment, and undertaking complicated

*The first two authors equally contributed to this work.
A preliminary version of this article was presented at *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'16)* [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' address: Y. Liu, Y. Jin, and P. Li, Department of Electrical and Computer Engineering, Texas A&M University, 400 Bizzell St., College Station, TX, 77843, USA; emails: {yliu129, jyyz, pli}@tamu.edu.

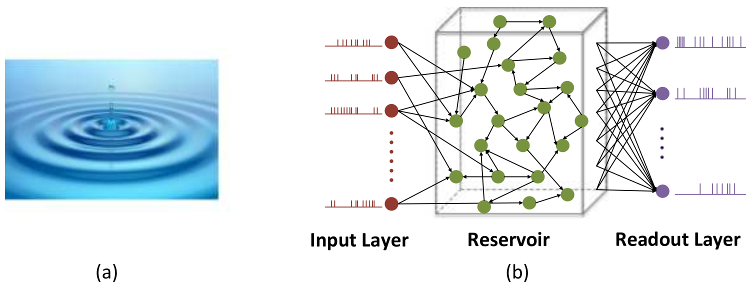


Fig. 1. (a) Dynamics in a liquid pond and (b) the liquid state machine.

decision-making problems with great energy and space efficiency. Therefore, the information-processing and communication patterns in the nervous system offer promising models for building the next-generation computing systems to address the performance and energy challenges currently faced by the computing industry.

The past decades have witnessed an endeavor to develop brain-like computers in both academia and industry. Recently, by using a cascade of many layers of nonlinear processing units, deep-learning algorithms—particularly convolutional neural networks (CNNs) [15] and deep neural networks (DNNs) [9]—have achieved tstate-of-the-art performance in a wide range of application, including image classification [13], natural-language processing [6], and handwriting recognition [5]. However, in order to deliver human-level performance on these deep networks, enormous amounts of resources and training efforts are required.

Significant research efforts have been dedicated to exploring biologically realistic spiking neural networks, which are anticipated to be power efficient and closely resemble brain behavior. There has been increasing interest in the concept of reservoir computing, which provides a computational model for exploiting the capability of recurrent neural networks [16, 18]. The liquid state machine (LSM) is one specific form of reservoir computing. As shown in Figure 1(b), the LSM consists of a reservoir, a set of randomly connected spiking neurons that models the complex recurrent topologies of cortical microcircuits, and a readout layer that receives reservoir responses. In the conventional LSM model, the reservoir synapses are fixed to relax the challenges of training. The rich high-dimensional dynamics created in the recurrent reservoir by spike inputs is very much like the long-lasting ripples triggered by throwing a pebble into a still pond (Figure 1(a)). With those rich dynamics as inputs, a linear readout layer can be trained for the final classification decision. The LSM is especially competent for classifying spatiotemporal patterns, such as speech recognition [8, 11, 25, 30].

It was only recently that silicon-based spiking neural systems (SNNs) have started to emerge, for instance, the NeuroPipe-Chip as a part of an accelerator board [22], the Neurogrid mixed-analog-digital multichip system [2], and IBM's TrueNorth chip [1]. However, those systems lack integrated on-chip learning capabilities, in general. While SNNs hold a lot of promise due to their closer resemblance to biological brains than older generations of artificial neural networks, the training of SNNs is not well understood at this point. It is a challenge to develop robust gradient-based training for SNNs, particularly recurrent SNNs.

To this end, the LSM is considered to be a good trade-off between the ability in tapping the computational power of recurrent SNNs and engineering tractability. Several recent works have investigated efficient hardware implementation of the LSM [12, 26, 27], with integrated efficient bio-inspired on-chip spike-dependent learning mechanisms to tune the reservoir and readout layer. For example, Zhang et al. [30] proposed a supervised probabilistic readout tuning algorithm, Jin and Li [10] proposed a stochastic activity-based spike-timing dependent plasticity (STDP) for tuning the reservoir, and Jin et al. [12] introduced a look-up table (LUT)–based STDP approach for efficient low-resolution tuning of the reservoir in hardware. Structural plasticity-based online reservoir optimization was proposed in [21]. Roy et al. [20] presented the hardware implementation of the LSM using analog circuitry with the readout trained by a gradient descent algorithm. Runtime energy management of LSM processors has also been examined from the aspects of runtime programmable arithmetic precision, activity-driven power gating, and activity-dependent reconfiguration [12, 27].

A preliminary version of the work in this article has been presented in [12]. In this article, we systematically analyze opportunities for enhancing energy efficiency while maintaining good learning performance of LSM neural processors from algorithmic and hardware implementation perspectives and further improve our neural processor architecture by addressing several key issues in terms of power efficiency.

The first key ingredient of the proposed energy-efficient self-adaptive LSM architecture is the exploration of a hardware-efficient STDP mechanism, which is motivated by two considerations. On the one hand, while the recurrent reservoir is in general difficult to train, the unsupervised and local nature of STDP makes it well suited for on-chip reservoir training, supplementing the training of the readout layer and improving learning performance [10, 14, 19, 29]. On the other hand, it is observed that STDP can naturally lead to a sparse recurrent reservoir network during the training process. This attractive self-organizing behavior is explored as an opportunity for runtime energy reduction.

While STDP is amenable to hardware realization, realizing STDP in a digital architecture with low overhead poses a substantial challenge. A straightforward implementation with high resolution introduces high hardware overhead, while utilizing low-bit resolution by sparsely sampling the STDP curve could harm the learning performance. To tackle the above problems, we propose the data-driven STDP that specifically targets efficient low-resolution hardware realization by minimizing the discretization error and simplifying the hardware design.

The second key ingredient of the proposed LSM processor is the incorporation of a novel runtime correlation-based neuron gating scheme to reduce the power dissipated by a large number of reservoir neurons. Under the context of the liquid state machine, the rich dynamics in the reservoir is typically crucial for achieving good learning performance. Consequently, energy consumed by reservoir neurons greatly contributes to the overall energy consumption of LSM neural processors. Our proposed correlation-based neuron gating mechanism monitors the correlated firing activities across the reservoir and deactivates neurons whose firing events are highly correlated with their presynaptic neuron(s) during runtime. This approach notably saves energy without any dramatic performance degradation.

The third technique presented in this article addresses energy inefficiency due to the memory-intensive nature of neural computation. Neural processors, including ones that are under consideration, generally require a large number of memory resources for storing various parameters, such as synaptic weights and internal neural states. Those memory elements heavily load the clock distribution network and their clock-induced toggling activities take a significant portion of the total power dissipation. We explore the regularity in the hardware structure and the process flow of the proposed LSM processors, which provide well-defined boundaries to partition the memory elements inside each neuron that are activated at different phases of neural processing. This leads to an activity-based clock gating mechanism with a granularity of a partitioned memory group inside each neuron.

We benchmark our proposed LSM through two real-world tasks (i.e., speech recognition and image classification) by using an adopted TI46 speech corpus and CityScape urban scenes dataset.

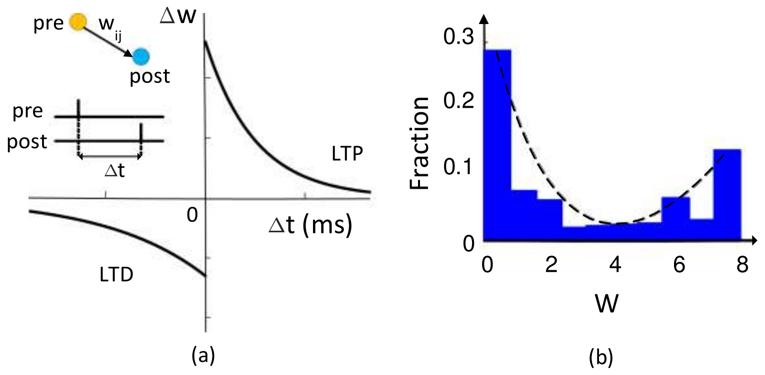


Fig. 2. (a) A typical STDP curve and (b) an equilibrium synaptic weight distribution obtained by the STDP rule in a reservoir.

It has been demonstrated that the proposed LSM neural processor boosts the average learning performance by up to 2.0% while reducing energy dissipation by up to about 29% compared to a baseline LSM design with little additional hardware overhead on a Xilinx Virtex-6 FPGA.

2 ALGORITHMIC LEVEL OPTIMIZATION

In this section, we introduce the hardware-friendly STDP for reservoir tuning and the novel correlation-based neuron gating method for energy reduction.

2.1 STDP for Reservoir Tuning

The STDP is a bio-inspired, unsupervised Hebbian learning mechanism realizing synaptic plasticity based on the respective spiking timing orders of the presynaptic and postsynaptic neurons [3]. The potentiation of the synapse w_{ij} happens when the presynaptic neuron j fires before the postsynaptic neuron i. The reversed firing order (i.e., a presynaptic neuron fires after the postsynaptic neuron) incurs the depression. The amount of weight modification Δw_{ij} relies on the temporal difference $\Delta t = t_{post} - t_{pre}$ between each presynaptic and postsynaptic spike pair:

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

where Δw^{+} and Δw^{-} are the weight updates caused by long-term potentiation (LTP) and long-term depression (LTD), and $A_{\pm}(w)$ determines the strength of LTP/LTD, respectively. Typical STDP characteristics are plotted in Figure 2(a).

Tuning a reservoir using STDP leads to two benefits: (1) a potential performance boost via the self-adaptation of recurrent connections, which we will show in Section 6 with experimental results; and (2) a refined sparse topology of the reservoir, which is exploited by us to build an energy-efficient hardware processor. To show the obtained sparse structure, we apply standard STDP and plot the converged synaptic weight distribution of the reservoir (Figure 2(b)). Generally, as a common setting, only excitatory synapses in the reservoir are assumed to be changeable and adjusted with STDP, whereas inhibitory synapses are fixed for the sake of stabilizing network dynamics. Here, the resulting bimodal weight distribution indicates a considerable amount of zero-valued and low-valued synapses, which can be turned off to save power without significantly impacting recognition performance.

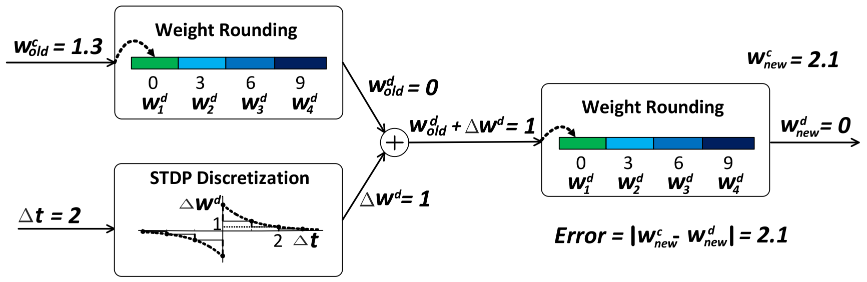


Fig. 3. The naïvely realized digital STDP.

A cost-effective realization of a given STDP on a digital processor presents an interesting challenge. A straightforward hardware implementation with high digital resolution closely approximates continuous STDP computation thus attains good performance boost. However, this approach results in a formidable cost, as all reservoir synaptic modules will integrate the high-precision STDP curve. The straightforward realization of (1) also incurs infinitely small weight modifications as the temporal difference Δt increases, not to mention that the number of such updates can be huge. Innumerable synaptic events with tiny weight changes will be triggered by these two effects, leading to high levels of cost and power inefficiency of hardware LSM processors. On the other hand, solely cutting hardware overhead by using a low resolution is likely to encounter an immediate performance drop.

2.2 Hardware-Friendly STDP for Reservoir Tuning

Implementing STDP with extremely-low-bit resolution while achieving good learning performance is challenging. To this end, we first restrict the STDP activation time window by setting a limit Δt_l the spike temporal difference Δt such that no synaptic update is considered when $\Delta t > \Delta t_l$. In this way, we can reduce the number of small-valued weight updates. Furthermore, to balance potentiation and depression, we add another constraint to that areas under LTD and LTP portion of the STDP curve are identical. With these two constraints, we still need to maintain the good performance of a delicately designed continuous STDP by properly discretizing both synaptic weights and the continuous STDP curve.

2.2.1 Naïve Digital STDP Realization. One intuitive realization of a B–bit STDP is uniformly discretizing the weight value into 2^B levels: $\{w^d_1, w^d_2, \ldots, w^d_{2^B}\}$ and similarly discretizing the weight change Δw^c of the continuous STDP rule within the activation window. In this and the following sections, the weight and weight change with superscript d stand for discretized values, while those with the superscript c represent the continuous ones. As illustrated in Figure 3, a spike timing difference Δt triggers a discretized synaptic change of Δw^d determined by the discretized STDP curve, which is then added up to the current (old) discretized weight w^d_{old} and rounded into the w^d_{new}:

$$\begin{aligned} w^d_{new} &= round(w^d_{old} + \Delta w^d(\Delta t)) \\ &= round(round(w^c_{old}) + \Delta w^d(\Delta t)), \end{aligned} \tag{2}$$

where $round(\cdot)$ rounds its argument to its nearest discretized level and w^c_{old} (w^c_{new}) is the current (new) continuous value if synaptic weights and STDP were implemented in real numbers.

A careful investigation of the above updating process uncovers *two key disadvantages*. On the one hand, an adder is required to perform each add operation (see Figure 3), introducing large

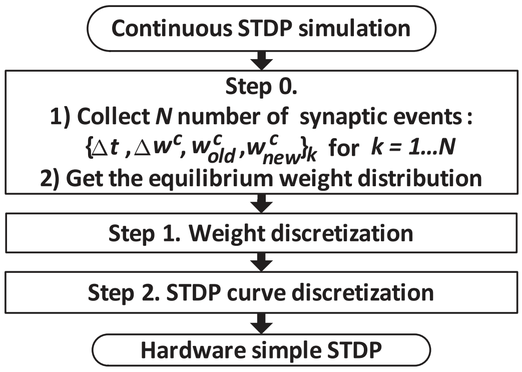

Continuous STDP simulation
Step 0.
1) Collect N number of synaptic events :
{Δt ,Δw^c, w^c_old, w^c_new}_k for k = 1...N
2) Get the equilibrium weight distribution
Step 1. Weight discretization
Step 2. STDP curve discretization
Hardware simple STDP

Fig. 4. The proposed data-driven offline STDP design flow.

hardware overhead. On the other hand, the computation of w^d_{new} in Figure 3 suffers from two types of rounding error: discretization of the real-valued weight w^c and quantization of the continuous weight update Δw^c. The specific example shown in Figure 3 explains this well. The continuous weight update (Δw^c) should be 0.8 when $\Delta t = 2$. Given the current weight ($w^c_{old} = 1.3$), the new continuous weight w^c_{new} is 2.1. However, the weight discretization rounds w^c_{old} down to $w^d_{old} = 0$ and the discretized weight update Δw^d is 1. Finally, the discretized updated weight $w^d_{old} + \Delta w$ is rounded to 0. Overall, the naïve discretization of STDP produces a very large quantization error of 2.1 under low-bit resolutions.

2.2.2 Proposed Data-Driven STDP Implementation. To minimize the above two aggregated quantization errors, our key innovation is to discretize the synaptic weight and the STDP curve collaboratively to match realistic synaptic events over a large set of STDP updates. The ideas behind our data-driven STDP design are: (1) discretizing the continuous weights such that the equilibrium weight distribution is well represented and (2) discretizing the STDP curve to match the characteristics of the synaptic update given the temporal difference Δt and weight change Δw^c. The proposed data-driven discretization mechanism shown in Figure 4 optimizes both synaptic weights and the STDP curve discretization. We formulate our design flow into steps and describe them in detail.

Step 0: Profiling the continuous STDP. Simulate the reservoir with typical inputs and collect synaptic events as sets of four values: $\{\Delta t_k, \Delta w^c_k, w^c_{old,k}, w^c_{new,k}\}, k \in [1, N]$. The converged weight distribution under the STDP is recorded as well.

Step 1: Characterizing the continuous weights. The goal here is to adequately represent the continuous weight distribution with a carefully chosen weighting scheme of B-bit integers. Therefore, to obtain the optimal discretization scheme $\{\hat{w}^d_j\}$ ($j = 1, \ldots, 2^B$), we want to minimize the representation error of each continuous weight w^c_i given 2^B unique weight levels over all collected weights:

$$\begin{aligned} \underset{w^d_j}{\text{minimize}} \quad & \sum_i \min_{w^d_j} \{(w^c_i - w^d_j)^2\} \\ \text{subject to} \quad & w^d_j \in [w_{min}, w_{max}], \forall j \in [1, \ldots, 2^B], \end{aligned} \tag{3}$$

where w^d_js are the digitized weight values and $\min_{w^d_j}\{(w^c_i - w^d_j)^2\}$ is the squared representation error for the ith collected continuous weight under the current discretization scheme.

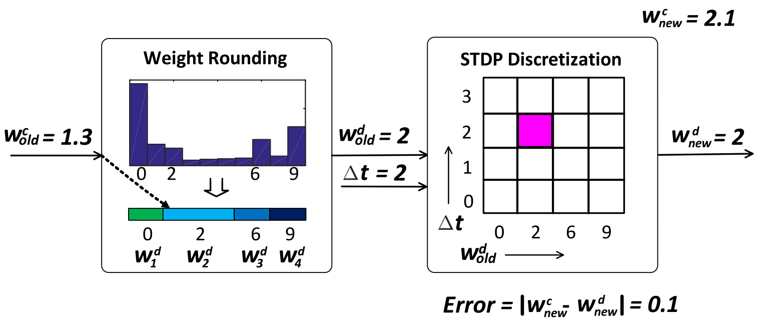


Fig. 5. The weight updating process of the proposed data-driven STDP.

Although solving (3) in a mathematically exact manner is extremely difficult, the optimization problem can be solved by exhaustive search since B is small.

Step 2: Discretizing the STDP curve. We optimize the discretized STDP rule based on the weighting levels obtained from Step 1. Given $\{\Delta t_k, w^d_{old,k}\}$ of the kth synaptic event, instead of discretizing the STDP curve to get the ΔW^d, our key idea is to map $\{\Delta t_k, w^d_{old,k}\}$ directly to the new weight value $w^d_{new,k}$ through an optimized LUT. This LUT is indexed by Δt and w^d_{old}, where the spike timing difference Δt has been already discretized by the properly chosen emulation timestep. Compared to the naïve approach in Figure 3, the proposed method brings down the hardware overhead by eliminating the add operation and merges multiple rounding steps into a single LUT, thereby avoiding provoking multiple digitization errors. Figure 5 demonstrates the weight update process of the proposed data-driven STDP using the same example of Figure 3 and shows that our approach results in a much smaller overall discretization error of 0.1.

Specially, the use of the LUT is to minimize the discretization error over the continuous STDP data collected in Step 0. Each LUT entry serves as a discretized resulting weight obtained under the proposed STDP rule. With the quantized weight levels w^d_js, we first map $\{\Delta t, w^c_{old}, w^c_{new}\}$ to $\{\Delta t, w^d_{old}, w^c_{new}\}$ for each recorded continuous synaptic event, where w^d_{old} is chosen to be the closest digitized weight level of w^c_{old}. In the LUT, this synaptic event is mapped to entry L_{mn} at the mth row and nth column, which is indexed by $\{\Delta t, w^d_{old}\}$ (see Figure 5). After this mapping is done for all collected data, each entry L_{mn} of the LUT now has its own set of the mapped continuous events, noted as $set(L_{mn})$. Our goal is to find an optimal value of each L_{mn} for discretizing w^c_{new} such that the aggregated error over all w^c_{new}s in $set(L_{mn})$ is minimized:

$$\begin{aligned} \underset{L_{mn}}{\text{minimize}} \quad & \sum_k (w^c_{new,k} - L_{ij})^2 \\ \text{subject to} \quad & L_{mn} \in \{w^d_1, w^d_2, \ldots, w^d_{2B}\} \\ & \{w^c_{new,k}\} \in set(L_{mn}). \end{aligned} \tag{4}$$

Essentially, the above optimal solution minimizes the summed squared root error for all continuous STDP updates that fall into a certain LUT entry. Again, this optimization problem can be easily solved offline due to the small design space.

2.3 Correlation-Based Neuron Gating

The rich dynamics in the reservoir is critical for good learning performance. However, the generation of such dynamically changing responses can be power consuming, since each active neuron

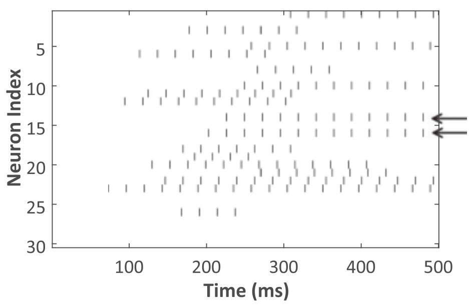


Fig. 6. A raster plot of the reservoir response. Only a part of the reservoir response is shown for simplicity. The firing events of two connected neurons 14 and 16 are highly correlated.

carries out a series of operations at each emulation timestep (see Section 3.2). Instead of randomly pruning reservoir neurons, which might result in a significant performance hit, we propose a novel runtime reservoir neuron gating approach based on the correlation between neuron firing activities with little impact on performance.

Our key observation is that two reservoir neurons or more may produce correlated firing activities, as depicted in the spike raster plot of Figure 6. Note that correlation between firing activities reveals redundancy among the corresponding neurons with respect to the objective of discriminating different input samples. In other words, a redundant neuron that replicates the spike train of another neuron does not contribute to the separability of different input patterns. In a pair of connected neurons, activities of the postsynaptic neuron may be identified to be highly correlated with the presynaptic one. If so, we bypass the computational steps of the postsynaptic neuron and set its spike output according to its presynaptic counterpart.

Implementing this approach in hardware entails efficient monitoring of correlation of firing activities, which we describe very briefly. For each pair of connected neurons, we compute the Hamming distance of the spike trains between the two neurons and sum it up over consecutive input samples as a measure of correlation:

$$\rho = \sum_{i=1}^{N} |s_p - s_q| (s_p, s_q \in \{0,1\}^n), \tag{5}$$

where s_p and s_q are the binary firing event sequences of length n of two connected neurons p and q, respectively, and N is the number of input samples. If the correlation measure ρ is smaller than a predefined threshold ρ_{th}, we consider the firing activities of two neurons as correlated. Section 4.2 describes hardware implementation of this correlation-based neuron-gating scheme in more detail.

3 HARDWARE IMPLEMENTATION ARCHITECTURE

In this section, we concisely describe the overall architecture of the LSM processors and the realization of digital spiking neurons in the network.

3.1 Overall LSM Processor Architecture

Figure 7 depicts the overall architecture of the LSM neural processor. The reservoir and the readout layer of Figure 1 are realized by a reservoir unit (RU) and a training unit (TU), respectively. Each reservoir (liquid) neuron is implemented with a liquid element (LE) in RU and the readout (output) neuron is implemented with the output element (OE). External input spikes are sent to

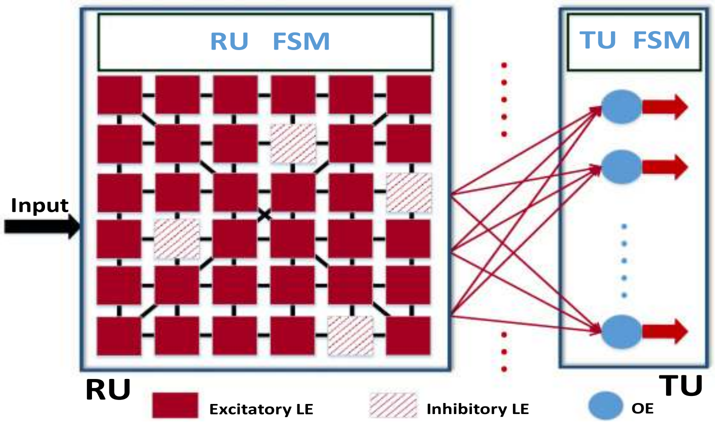


Fig. 7. Overall architecture of an exemplary LSM neural processor.

their targeted LEs through a predefined crossbar interface. All LEs receive and process input spikes in parallel, controlled by a global finite state machine (FSM) in the RU. The spikes generated from LEs are registered and sent to all OEs through fully connected readout synapses. The FSM in the TU controls the parallel operations of OEs. In addition, reservoir spikes are fed back to other LEs following certain connectivity patterns defined by another crossbar interface.

The learning process of an LSM processor is performed in three separate phases in time order: the reservoir training phase, the correlation-based gating phase, and the readout training phase. First, in the reservoir training phase, the RU is trained by the proposed STDP algorithm until the synaptic weight distribution converges. The gating phase then takes place, during which the reservoir neurons fix their synaptic weights, take input spikes, and count the occurrence of correlated presynaptic and postsynaptic responses throughout the phase. After all input patterns have been fed to the neural processor, the gating decision will be made inside each neuron based on the correlation-based neuron gating approach proposed in Section 2.3. At last, during the readout training phase, the TU is trained by a biologically plausible supervised spike-based algorithm [30] to perform the classification. The RU continues to be activated to provide spike inputs to the TU while maintaining its synaptic weights and gating decisions during the readout training phase.

3.2 Implementation of Digital Spiking Neurons

The proposed LSM neural processors operate through a series of computational steps and require a large number of storing elements inside each neuron. As shown in Figure 8, a digital neuron element (LE or OE) contains three major functional modules for neural computing and the LE has a correlation-based neuron-gating control module in addition. The unique architectural and functional properties of the proposed LSM neural processor naturally lead to well-defined boundaries between these modules in terms of execution and storage. At a certain emulation timestep, first, the synaptic input processing module computes the second-order synaptic spike responses with the arrival of spike inputs. Then, the action potential (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.

The neuron-gating control module in an LE is activated only during the gating phase. During the gating phase, at each emulation timestep, the gating module activates after the execution of the

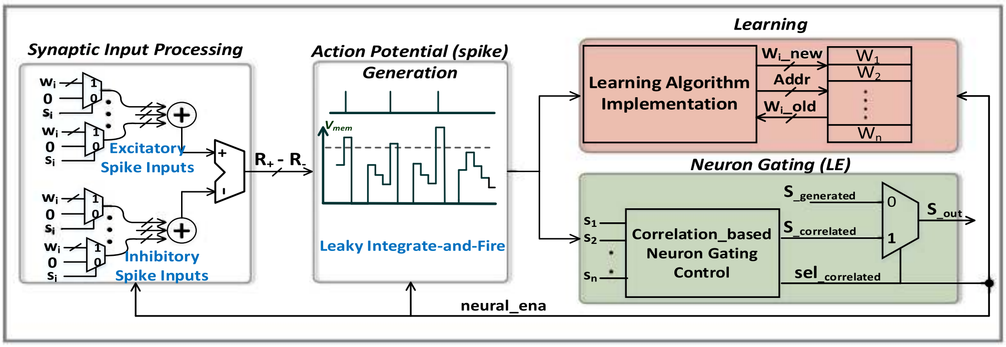


Fig. 8. Hardware implementation of a single digital neuron element (LE or OE). The neuron-gating module resides only in the LE.

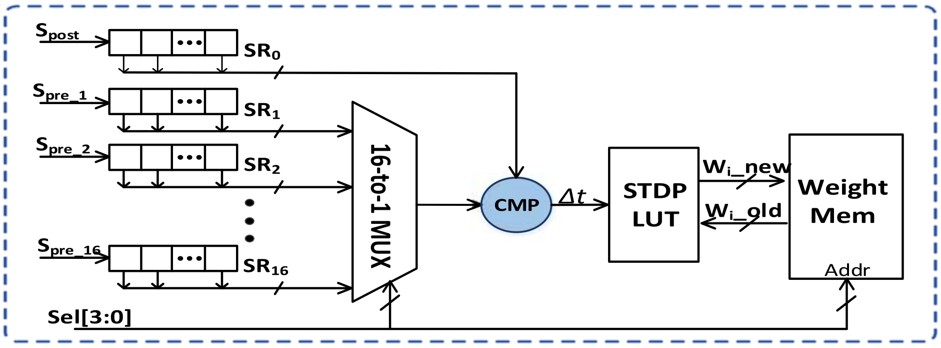


Fig. 9. Implementation of the proposed STDP learning algorithm.

spike-generation module to examine the correlation between spikes at two ends of each synapse and potentially turns off the three other modules within the same neuron during the readout training and the inference phase. Implementation details of correlation-based neuron gating will be described in Section 4.2.

LEs and OEs differ on learning module implementation in terms of learning functions, arithmetic resolutions of digital synapses, and the realization of weight storage. A block memory (BRAM on FPGAs) inside each OE is used to store all its presynaptic weights. LEs, on the other hand, make use of distributed RAMs with much smaller sizes, which are realized by the LUTs on FPGAs because of the lower synaptic bit resolution and the sparser connections in the reservoir.

4 CIRCUIT LEVEL OPTIMIZATION

In this section, we discuss the hardware implementation of the presented STDP mechanism, proposed correlation-based neuron-gating scheme, and tactivity-dependent clock gating for boosting the energy efficiency of the LSM processors.

4.1 Implementation of STDP

The learning unit in the LE implements the proposed hardware-friendly STDP reservoir-tuning mechanism, as depicted in Figure 9. We adopt the combinational logic STDP implementation proposed in [4] and further simplify it. In our proposed LSM architecture, the maximum fan-in of

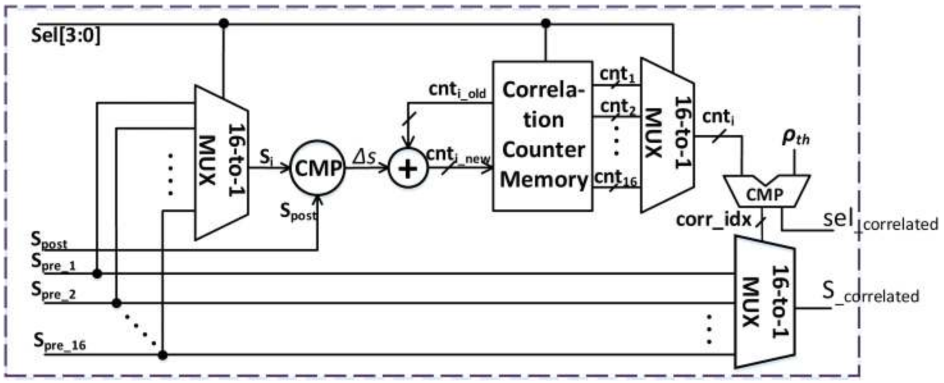


Fig. 10. Implementation of the correlation-based neuron-gating control logic.

each reservoir neuron is set to 16, as it results in a good learning performance while maintaining a reasonable connection density to avoid large overhead. The presynaptic shift registers (i.e., SR_1 to SR_{16}) keep track of the presynaptic spike events and the postsynaptic shift register (i.e., SR_0) is used for tracking firing events of the neuron itself where the STDP learning unit resides. The depths of presynaptic and postsynaptic shift registers represent time windows Δt_l for LTP and LTD, respectively, as defined in Section 2.2.

When a neuron fires, the generated spike is injected into its affiliated shift register from the MSB and shifted one bit to the right at every biological step of the neural system. The computation of Δt for pre- and postsynaptic spike pairs and the update of the associated synaptic weights are executed in serial. By examining the relative position of spikes in shift registers, the temporal difference Δt between presynaptic and postsynaptic spikes can be easily computed. For example, when a "1" appears on the MSB of SR_0, meaning that the postsynaptic neuron just fires, we check from SR_1 to SR_{16} to identify the presynaptic neurons that fire beforehand. If there is a presynaptic spike to be paired, the associated Δt is the location of this presynaptic spike in its shift register. Similarly, we can capture potential post-before-pre firing pairs if there are spikes appearing on the MSB of presynaptic shift registers. With Δt and the current weight value Wi_{old} of the corresponding synapse, the new synaptic weight Wi_{new} is directly acquired from the predefined STDP LUT.

4.2 Implementation of Correlation-Based Neuron Gating

Figure 10 depicts the correlation-based neuron-gating control unit inside the neuron-gating module in Figure 8. The gating phase starts after the reservoir-training phase and the reservoir synapses maintain the converged weight distribution. The correlation between the presynaptic and postsynaptic neuron is monitored whenever spike events appear on either end of an afferent synapse. In Figure 10, S_i represents one of the presynaptic spikes that is currently being checked while S_{post} is the postsynaptic spike. If the spike events differ on the two ends of a synapse, in other words, either the presynaptic or postsynaptic neuron fires while the other one does not, the comparison of S_i with S_{post} in leads to a logic "1" for Δs. Otherwise, a logic "0" is produced. The comparison result Δs is then added to the current value of the corresponding correlation counter stored in the memory. After all input patterns have been fed to the neural processor, in each neuron, the correlation counters are compared with the correlation threshold ρ_{th} defined in Section 2.3 serially. If a correlation counter is found to be less than ρ_{th}, the gating-control signal $\text{sel}_{\_correlated}$ is set to 1, meaning that a correlated presynaptic neuron with the index *corr_idx* has been identified. At the same time, the spike output $\text{S}_{\_correlated}$ of the current neuron is wired to the presynaptic

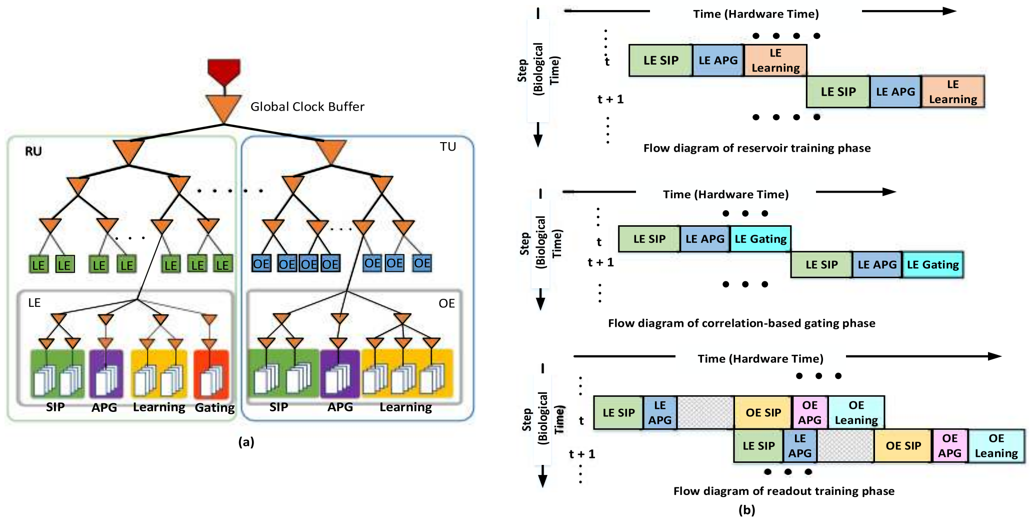


Fig. 11. (a) Clock distribution of the LSM. (b) SIP: Synaptic Input Processing, APG: Action Potential (Spike) Generation.

spike of the identified presynaptic neuron. With one correlated presynaptic neuron found, other correlation counters in the same neuron are not checked.

During the readout training and inference phase, a neuron that correlates with one of its presynaptic neurons is gated off in the sense that the three major functional modules—i.e., synaptic input processing module, action potential (spike) generation module, and learning module—are powered off. As mentioned before, the correlated presynaptic spike input is fed to the output of this neuron.

4.3 Implementation of Activity-Dependent Clock Gating

Neural processors, including ones that are targeted, are typically memory intensive. The large amount of storage spanning across the design heavily loads the clock distribution network and their clock-induced toggling activities take a significant portion of the total power dissipation. On the FPGA platform, for example, with a global clock driving extensive registers and on-chip memories through a dedicated clock tree, more than 60% of the total processor dynamic power would be dissipated by the clock tree and switching activities of the registers and memories.

To this end, we recognize that the architectural and functional regularity of the proposed LSM processor mentioned in Section 3.2 provides well-defined boundaries within which storage elements reside. Each stage in the neural process flow (Figure 11(b)) corresponds to a module inside a neuron element shown in Figure 8, which is active only during its corresponding stage. As shown in Table 1, the four processing stages in LE and three processing stages in OE take various numbers of clock cycles and involve different subsets of the registers and memories. The nature of the proposed LSM processors allows us to partition the on-chip storage in each neuron into different groups that are activated at different stages, leading to a fine-grained activity-dependent clock gating at the granularity of memory elements inside each neuron.

Figure 11(a) illustrates the clock distribution of the proposed LSM processor architecture. As shown in the figure, memory elements inside each neuron are driven by leaf nodes of the clock tree. On the FPGA, which is chosen as our demonstration platform, dedicated routing resources

Table 1. Numbers of FSM States, Memory Element Bits, and Cycle Occupancies Inside Neurons

	# of States	# of Memory Bits	Stage	Clock Cycles	Active Bits
LE	14	247	Synaptic Input Processing	49	40
			Action Potential Generation	3	11
			Learning	32	36
			Neuron Gating	80	160
OE	10	1,166	Synaptic Input Processing	271	64
			Action Potential Generation	3	13
			Learning	405	1,089

are responsible for distributing clock signals to ensure that they are delivered across the design in low skew. In this circumstance, directly gating the clock signal may jeopardize the low-skew performance ensured by the dedicated clock routing since it involves unconstrained flip flops and LUTs. With this constraint in mind, instead, we lower the clock power contribution by utilizing clock-enable (CE) signals to reduce the clock-triggered switching activities within memory elements. In each neuron, the memory elements inside the same module shown in Figure 11(a) share a common CE signal. If the memory elements are implemented with registers, this CE signal will be connected to the local CE signal of corresponding slices, which are the basic logic blocks of the FPGA. For the memory-implemented storage elements, the CE signals directly enable or disable the memory clock inputs. For both LEs and OEs, the activated stages of each module span across several well-defined global FSM states. Therefore, the activity-dependent CE signal of each module is encoded from the current state of the associated FSM.

One thing to mention is that the on-chip storage partitioning scheme is based on the unique architectural and functional characteristics of the proposed LSM processors and largely independent of the specific implementation platform. Therefore, the proposed activity-dependent clock gating technique can be exploited by an LSM processor in general and similar power benefits would be expected across different platforms. Moreover, the above clock-enabling approach does not reduce the power dissipated by the clock tree itself due to the limitation on FPGA platforms. Since ASIC implementations are not restricted by the aforementioned FPGA clock-routing constraints, direct gating on the clock signal may be added on top of the proposed activity-dependent clock-enabling approach to further optimize power consumption.

5 EXPERIMENTAL SETTINGS AND BENCHMARKS

Using the approaches described in [30], several digital LSMs are set up with different reservoir sizes and readout synaptic resolutions and simulated by the software simulator to fully judge the performance boost and sparsity resulting from the proposed STDP scheme. A 5-fold cross-validation scheme is adopted to assess learning performance. When doing the classification, the recognition decision is made by the LSM right after each testing sample is presented and the class label of the readout neuron with the highest firing rate is deemed to be the classification decision. In order to measure the impacts of proposed techniques on hardware overhead and energy consumption, we implement a representative design of the proposed LSM neural processor architecture with 135 reservoir neurons and 26 readout neurons targeted on the Xilinx Virtex-6 FPGA platform.

In order to thoroughly assess the learning performance and energy benefits of our proposed neural processor, we choose two nontrivial tasks of speech recognition and image classification. The first adopted real-world benchmark is a subset of the TI46 speech corpus [24], which contains

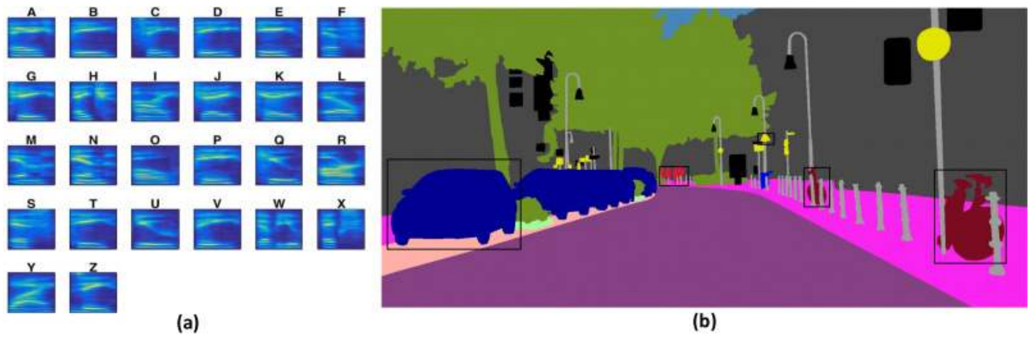


Fig. 12. (a) The spatiotemporal information of each speech generated by preprocessing. (b) A street scene of the CityScape dataset.

Table 2. The Identifiers of the Image Instances Extracted from the CityScape Dataset

Class ID	0	1	2	3	4	5	6	7	8
Object Name	Sidewalk	Building	Wall	Fence	Pole	Traffic Light	Traffic Sign	Vegetation	Terrain
Class ID	9	10	11	12	13	14	15	16	17
Object Name	Sky	Person	Rider	Car	Train	Motorcycle	Bicycle	Bus	Truck

ten spoken utterances of English letters from “A” to “Z.” There are 260 samples in this benchmark. The continuous temporal speech signals are preprocessed by Lyon’s ear model [17]; Figure 12(a) visualizes the input speech patterns acquired after the preprocessing stage. The preprocessed signals are then encoded into 78 spike trains using the BSA algorithm [23]. Each obtained input spike train is sent to 32 randomly selected reservoir neurons with a fixed weight randomly chosen to be 2 or -2; 26 readout neurons are required for this task. To the best knowledge of the authors, the best reported performance on the same benchmark is 94.6% [10], when another STDP-based reservoir tuning (i.e., AP-STDP) is used. However, we expect that the design complexity and the overhead to implement AP-STDP on the hardware can be too costly, as we have a heuristic reservoir synaptic weight update scheme and stop-learning condition in the algorithm. Therefore, we consider the proposed hardware-friendly STDP integrated on our LSM neural processor a better trade-off between learning performance and hardware implementation complexity.

Figure 12(b) illustrates the second benchmark that we adopted from the CityScape dataset [7], which contains images of the semantic urban scenes taken in several European cities. We select 18 different types of objects (see Table 2), segment them from the street scene and remap them into images of size 15×15. There are 60 instances for each labeled object, therefore 1080 images in total in the dataset. For a remapped image, 225 input spike trains are generated from a Poisson process with the probability proportional to the pixel value of each image. Each input spike train is connected to 4 randomly chosen reservoir neurons with a fixed weight to be 8 or -8 randomly. Since only 18 readout neurons are employed for this task, the remaining 8 OEs are turned off when this task is performed on the implemented hardware LSM processor.

6 EXPERIMENTAL RESULTS

Using the experimental settings in Section 5, we report the learning performance and the benefits brought by various optimization techniques in terms of hardware overhead and energy dissipation.

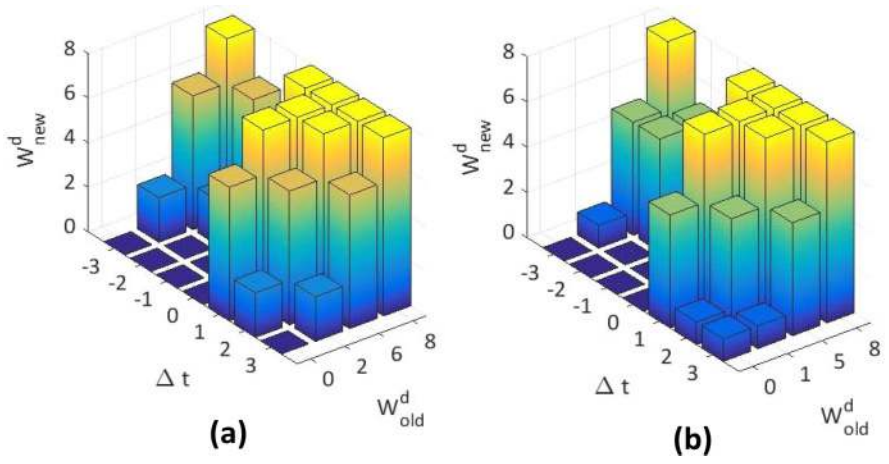


Fig. 13. The optimal STDP lookup tables for (a) spoken English letter recognition and (b) segmented image recognition.

Table 3. Performances of LSMs Without STDP Tuning

	Spoken Letter Recognition					
	Bit Resolution of Readout Synapses					
Reservoir Size	10	9	8	7	6	5
135	90%	90%	90.4%	88.8%	89.2%	89.2%
90	87.7%	86.5%	85.8%	86.9%	84.2%	84.2%
72	84.2%	83.1%	83.5%	82.7%	82.7%	80.4%
63	86.9%	86.2%	88.1%	85.9%	85.8%	82.7%
45	80.8%	78.8%	79.2%	80.8%	78.5%	72.7%

	Segmented Image Recognition					
	Bit Resolution of Readout Synapses					
Reservoir Size	10	9	8	7	6	5
135	96.6%	96.5%	96.6%	96.6%	96.4%	95.6%
90	96.0%	96.1%	95.9%	95.9%	95.2%	95.2%
72	94.9%	94.9%	95.2%	95.1%	95.3%	94.9%
63	93.7%	93.6%	93.8%	93.4%	93.5%	92.3%
45	92.7%	92.4%	92.5%	92.6%	92.2%	90.9%

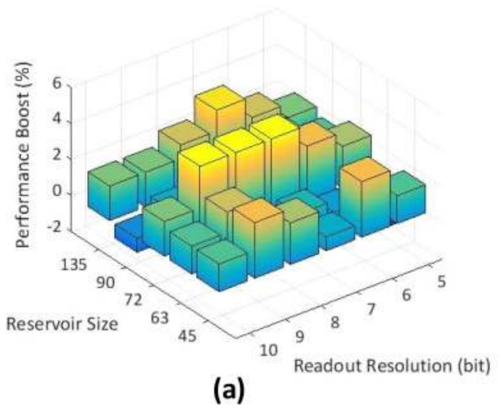
6.1 Performance Boost of the Hardware STDP

We adopt the optimized continuous STDP curve from [12]. To realize a low-cost hardware implementation, we digitize the reservoir synaptic weights with 2-bit resolutions and only consider the weight changes with $|\Delta t| \leq 3$. Following the proposed offline design flow introduced in Section 2.2.2, we collect a set of synaptic event data for two adopted benchmarks and design the optimized hardware-friendly STDP. Figure 13(a) and Figure 13(b) show the optimal weight discretization levels and visualize the STDP lookup table for the two applications, respectively.

Given the considered design space, the recognition performances of several digital LSMs without and with the STDP reservoir tuning are reported in Table 3 and Table 4, respectively. It shows that the best performance of the LSM neural processor without STDP tuning is 90.4%

Table 4. Performances of LSMs with STDP Tuning

	Spoken Letter Recognition					
	Bit Resolution of Readout Synapses					
Reservoir Size	10	9	8	7	6	5
135	91.9%	91.9%	93.1%	92.7%	91.8%	91.2%
90	86.9%	87.3%	88.1%	88.1%	86.5%	85.7%
72	86.1%	87.3%	87.7%	86.9%	85.8%	82.7%
63	88.5%	88.8%	88.5%	86.9%	86.2%	81.2%
45	82.3%	81.9%	81.5%	81.5%	81.5%	74.2%
	Segmented Image Recognition					
	Bit Resolution of Readout Synapses					
Reservoir Size	10	9	8	7	6	5
135	97.5%	97.9%	97.7%	97.4%	97.2%	96.8%
90	97.3%	97.0%	97.1%	96.9%	96.8%	96.0%
72	96.8%	96.5%	96.8%	96.6%	96.9%	95.6%
63	94.9%	95.4%	95.1%	94.8%	95.0%	93.4%
45	93.8%	93.9%	94.0%	93.9%	93.2%	92.2%



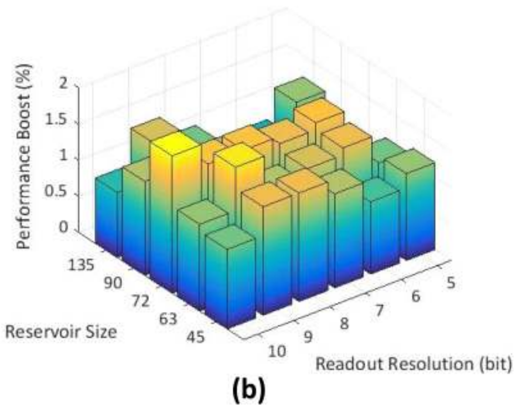


Fig. 14. The performance boosts for (a) spoken English letter recognition and (b) segmented image recognition by the proposed STDP scheme over the large design space.

for speech recognition and 96.6% for image recognition. When the STDP tuning applies, the best performances obtained are improved to 93.1% and 97.9%, respectively. Figure 14 visualizes the performance boosts achieved by the proposed STDP tuning scheme.

To better illustrate the power of the proposed hardware-friendly STDP mechanism, we compare it with the naïve discretized STDP approach mentioned in Section 2.2.1 and three other LUT-based rules (C1 to C3), in which the LUT entries are filled randomly. The averaged performance boost achieved by all considered rules are reported in Figure 15. The results show that the proposed STDP produces an average performance boost of 2% and 1.2% for speech and image recognition, respectively, outperforming all other STDP rules.

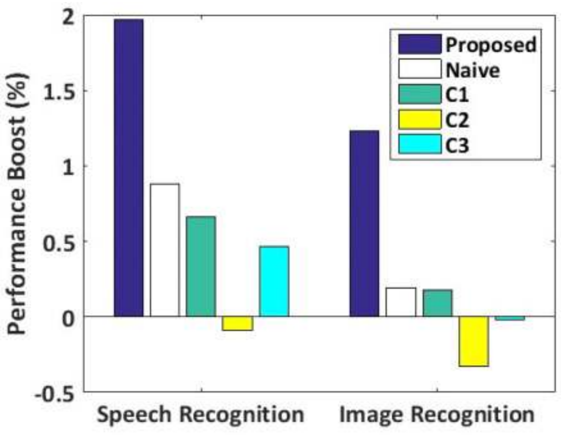


Fig. 15. The average performance boosts over the considered design space attained by different STDP rules.

Table 5. The Reservoir Synaptic Reductions of the Proposed STDP

	Spoken Letter Recognition				
Reservoir Size	**135**	**90**	**72**	**63**	**45**
Reduction	27.2%	28.9%	28.7%	29.2%	27.5%
	Segmented Image Recognition				
Reservoir Size	**135**	**90**	**72**	**63**	**45**
Reduction	21.9%	20.2%	22.6%	23.9%	18.0%

6.2 Sparsity of the Proposed Architecture

In our work, the reservoir sparsity achieved by the proposed STDP rule is measured by the percentages of zero-valued synaptic weights after reservoir tuning. The results are shown in Table 5 of LSM processors with various reservoir sizes.

With the proposed reservoir-tuning scheme applied, we examine recognition performance boosts compared to the baseline of a representative LSM processor with 135 reservoir neurons and 10-bit readout resolution. Performance boosts with different numbers of bypassed reservoir neurons are plotted in Figure 16. As seen here, up to 30% of reservoir neurons whose activities are correlated can be powered off without any dramatic drop in performance, which potentially helps to improve the overall energy efficiency.

6.3 Hardware Overhead and Energy Consumption of the Proposed Architecture

To illustrate the impacts on hardware overhead and energy consumption of the techniques mentioned in this article, we implement three LSM neural processors that incorporate different combinations of the proposed energy optimization techniques. Among them, the "baseline LSM" design serves as a reference that is constructed with a fixed reservoir and does not implement any energy optimization technique; the "adaptive LSM" design integrates the hardware-friendly STDP to form an adaptive reservoir and the activity-dependent clock-gating; the "adaptive LSM with correlation-based gating" neural processor incorporates all three techniques described in this article. Table 6 shows the comparison of hardware resource utilization 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. It is evident that with the implementation of proposed techniques, we still

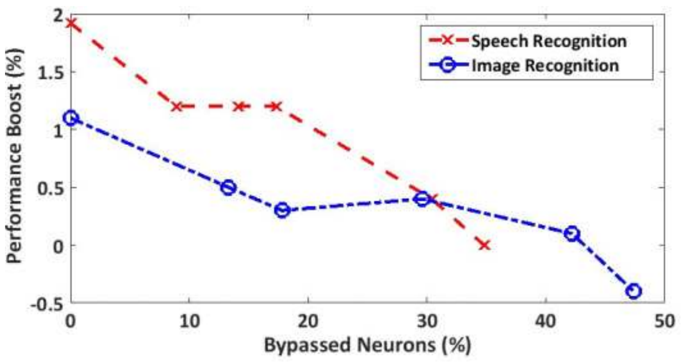


Fig. 16. The performance boosts achieved by the proposed STDP under different levels of gated correlated neurons.

Table 6. Comparison on Hardware Resource Utilization

	FFs	LUTs	FF Percentage of Usage	LUT Percentage of Usage
Baseline LSM	10519	40274	3%	26%
Adaptive LSM	10920 (+3.8%)	48419 (+20.2%)	3%	32%
Adaptive LSM with Correlation-Based Gating	10938 (+4.0%)	50317 (+24.9%)	3%	33%

have efficient hardware utilization with respect to the available resources on the targeted FPGA board.

Based on activity-based simulation data, we compare the dynamic power of the three neural processors clocked at 100MHz in Table 7. The adaptive LSM with a correlation-based gating neural processor has 20% reservoir neurons gated, which boosts performance noticeably, by up to 1.2% over the baseline. Since the baseline LSM processor has a fixed reservoir, the reservoir-training phase does not apply to it. Note that the power consumption of a correlation-based gating phase itself is nonnegligible. However, applying the correlation-based gating largely benefits readout training and classifying power. Considering that readout training takes the majority of training time, total training energy will be significantly reduced with a smaller readout training power, which is demonstrated in Table 8.

Using the power data from Table 7, we calculate energy consumption for training and classifying a representative sample (a speech sample of one letter in spoken letter recognition and an image of one segmented object instance in segmented image recognition) of the three LSM neural processors. To get good learning performance, 25 epochs of reservoir training and 250 epochs of readout training are conducted for each sample in both applications. The correlation-based gating and the inference phase are executed for only one iteration. The training energy is the sum of the energies consumed for the reservoir training, correlation-based gating and the readout training stages.

From Table 8, it is clear that the cooperation of three techniques introduced in this article can effectively reduce the energy consumption of the LSM neural processor considerably. The results

Table 7. Comparison on Average Power Dissipation (Unit: mW)

	Spoken Letter Recognition			
	Reservoir Training	**Correlation Gating**	**Readout Training**	**Classifying**
Baseline LSM	/	/	232	249
Adaptive LSM	224	/	186	209
Adaptive LSM with Correlation-Based Gating	226	362	166	185
	Segmented Image Recognition			
	Reservoir Training	**Correlation Gating**	**Readout Training**	**Classifying**
Baseline LSM	/	/	242	246
Adaptive LSM	222	/	196	193
Adaptive LSM with Correlation-Based Gating	214	387	169	170

Table 8. Comparison on the Energy Consumed for Training and Classifying a Single Speech or Image Sample (Unit: mJ)

	Spoken Letter Recognition	
	Training	**Classifying**
Baseline LSM	269.51	0.53
Adaptive LSM	216.71(−19.6%)	0.44(−16.9%)
Adaptive LSM with Correlation-Based Gating	194.09(−28.0%)	0.39(−26.4%)
	Segmented Image Recognition	
	Training	**Classifying**
Baseline LSM	246.26	0.46
Adaptive LSM	202.78(−17.6%)	0.36(−21.7%)
Adaptive LSM with Correlation-Based Gating	175.42(−28.8%)	0.32(−30.4%)

have shown that the proposed LSM neural processor is up to 29% more energy efficient for training and 30% more energy efficient for classifying than the baseline.

We are aware that the Xilinx design tools offer a standard intelligent clock gating in general [28] by preventing logic not used in a given clock cycle from toggling. To better illustrate the energy efficiency of the proposed clock-gating approach, we apply the standard clock gating provided by Xilinx ISE and our proposed activity-dependent clock gating, respectively, on top of the LSM processor that has the adaptive reservoir and the correlation-based gating scheme and compare

Table 9. Comparison of the Energy Reduction of Standard Clock Gating and the Proposed Clock Gating

	Spoken Letter Recognition	
	Training	**Classifying**
Standard Clock Gating	227.31	0.45
Proposed Clock Gating	194.09	0.39
	Segmented Image Recognition	
	Training	**Classifying**
Standard Clock Gating	213.76	0.34
Proposed Clock Gating	175.42	0.32

Note: Both designs have a trainable reservoir and correlation-based neuron gating (Unit: mJ).

the energy results in Table 9. The results show that our proposed clock gating outperforms the standard clock gating in energy efficiency. It is reported that the clock gating implemented by the Xilinx tool applies CE signals only to the weight storage elements (i.e., weight registers in LEs and BRAMs in OEs), which suggests that the unique regularities of the LSM architecture are not recognized and exploited. In comparison, the proposed clock-gating method takes full advantage of the unique architectural and functional properties of the LSM processor and implements fine-grained CE signals for all storage elements in each neuron.

7 CONCLUSION

In this article, we propose an energy-efficient LSM processor with several novel optimization techniques from both algorithmic and hardware design perspectives. Among these, a hardware-friendly STDP approach is presented for refining the reservoir and boosting learning performance; a correlation-based neuron-gating scheme enables online reconfiguration for improving energy efficiency with little performance degradation; and an activity-dependent clock gating is leveraged for additional energy saving. Using two different types of real-world applications to benchmark, we have shown that the proposed LSM processors achieve efficient reservoir tuning with low-bit resolution, deliver good learning performance, and outperform the baseline design in terms of learning performance and energy efficiency on an FPGA platform.

REFERENCES

[1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, and others. 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, 1537–1557.

[2] 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. *Proceedings of IEEE* 102, 5, 699–716.

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

[4] Andrew Cassidy, Andreas G. Andreou, and Julius Georgiou. 2011. A combinational digital logic approach to STDP. In *IEEE International Symposium on Circuits and Systems (ISCAS'11)*. IEEE, 673–676.

[5] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. 2010. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation* 22, 12, 3207–3220.

[6] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, 160–167.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*.

[8] 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.

[9] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18, 7, 1527–1554.

[10] Yingyezhe Jin and Peng Li. 2016. AP-STDP: A novel self-organizing mechanism for efficient reservoir computing. In *International Joint Conference on Neural Networks (IJCNN'16)*. IEEE, 1158–1165.

[11] Yingyezhe Jin and Peng Li. 2017. Performance and robustness of bio-inspired digital liquid state machines: A case study of speech recognition. *Neurocomputing* 226, 145–160.

[12] Yingyezhe Jin, Yu Liu, and Peng Li. 2016. SSO-LSM: A sparse and self-organizing architecture for liquid state machine based neural processors. In *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'16)*. IEEE, 55–60.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[14] Andreea Lazar, Gordon Pipa, and Jochen Triesch. 2009. SORN: A self-organizing recurrent neural network. *Frontiers in Computational Neuroscience* 3, 23.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of IEEE* 86, 11, 2278–2324.

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

[17] Richard F. Lyon. 1982. A computational model of filtering, detection, and compression in the cochlea. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (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, 2531–2560.

[19] David Norton and Dan Ventura. 2006. Preparing more effective liquid state machines using Hebbian learning. In *International Joint Conference on Neural Networks (IJCNN'06)*. IEEE, 4243–4248.

[20] Subhrajit Roy, Amitava Banerjee, and Arindam Basu. 2014. Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations. *IEEE Transactions on Biomedical Circuits and Systems* 8, 5, 681–695.

[21] Subhrajit Roy and Arindam Basu. 2016. An online structural plasticity rule for generating better reservoirs. *Neural Computation* 28, 11 (2016), 2557–2584.

[22] Tim Schoenauer, Sahin Atasoy, Nasser Mehrtash, and Heinrich Klar. 2002. NeuroPipe-Chip: A digital neuro-processor for spiking neural networks. *IEEE Transactions on Neural Networks* 13, 1, 205–213.

[23] 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.

[24] TI46. The TI46 Speech Corpus. Retrieved November 11, 2017 from 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. *Information Processing Letters* 95, 6, 521–528.

[26] Qian Wang, Yingyezhe Jin, and Peng Li. 2015. General-purpose LSM learning processor architecture and theoretically guided design space exploration. In *IEEE Biomedical Circuits and Systems Conference (BioCAS'15)*. IEEE, 1–4.

[27] 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 *IEEE International Symposium of Circuits and Systems (ISCAS'16)*. IEEE, 361–364.

[28] Xilinx. Xilinx Intelligent Clock Gating. Retrieved November 11, 2017 from https://www.xilinx.com/support/documentation/application_notes/xapp790-7-series-clock-gating.pdf.

[29] Fangzheng Xue, Zhicheng Hou, and Xiumin Li. 2013. Computational capability of liquid state machines with spike-timing-dependent plasticity. *Neurocomputing* 122, 324–329.

[30] 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, 2635–2649.

Received April 2017; revised July 2017; accepted September 2017