Exploring Sparsity of Firing Activities and Clock Gating for Energy-Efficient Recurrent Spiking Neural Processors

Yu Liu
Department of Electrical and Computer Engineering
Texas A&M University
Email: yliu129@tamu.edu

Yingyezhe Jin
Department of Electrical and Computer Engineering
Texas A&M University
Email: jyyz@tamu.edu

Peng Li
Department of Electrical and Computer Engineering
Texas A&M University
Email: pli@tamu.edu

***Abstract*—As a model of recurrent spiking neural networks, the Liquid State Machine (LSM) offers a powerful brain-inspired computing platform for pattern recognition and machine learning applications. While operated by processing neural spiking activities, the LSM naturally lends itself to an efficient hardware implementation via exploration of typical sparse firing patterns emerged from the recurrent neural network and smart processing of computational tasks that are orchestrated by different firing events at runtime. We explore these opportunities by presenting a LSM processor architecture with integrated on-chip learning and its FPGA implementation. Our LSM processor leverage the sparsity of firing activities to allow for efficient event-driven processing and activity-dependent clock gating. Using the spoken English letters adopted from the TI46 [1] speech recognition corpus as a benchmark, we show that the proposed FPGA-based neural processor system is up to 29% more energy efficient than a baseline LSM processor with little extra hardware overhead.**

I. Introduction

Microcircuits in the brain feature the generation of action potentials, or spikes, and complex recurrent network topologies. The recently emerged concept of reservoir computing has received increased attention as it provides a computational model for exploiting the power of recurrent neural networks [2]. As a model of reservoir computing, the liquid state machine (LSM) operates upon spiking neurons and is especially competent for spatiotemporal pattern classification such as speech recognition and bio-signal processing [3]–[5]. Structurally speaking, the LSM is composed of a reservoir consisting of spiking neurons wired up to form a recurrent network and a readout layer receiving inputs from the reservoir as in Fig. 1. In the standard LSM model, the synaptic weights of reservoir neurons are fixed to avoid the difficulty in training. Via its nonlinear dynamics, the reservoir maps input patterns to higher-dimensional transient responses, which are fed to readout neurons for final classification through the plastic synapses projecting from the reservoir to the readout layer.

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.

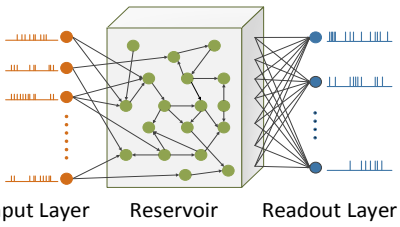

Fig. 1. The model of the liquid state machine.

Spiking neural networks (SNNs) in general have been targeted for VLSI realization recently [6]–[8], however, without on-chip learning capability. This may be explained by two reasons. While holding a lot of promise due to their closer resemblance to biological brains than older generations of artificial neural networks, SNNs are difficult to train. In general, the problem of training SNNs is not well understood at this point. On the other hand, naive gradient-based learning mechanisms do not work well for SNNs and their hardware implementations can lead to high design complexity and excessive hardware and energy overhead.

To this end, the LSM is envisioned as a good trade-off between the ability in tapping the computational power of recurrent spiking neural networks and readiness for training. Recently, the unique architectural properties of the LSM have been leveraged for cost-effective hardware implementations [9], [10]. As part of it, interesting bio-inspired spike-dependent training algorithms for both the reservoir and the output layer have emerged for LSM neural processors, for example, a supervised probabilistic readout tuning algorithm [5],a probabilistic spike-based rule with stopping learning [11], and a hardware-optimized spike-timing-dependent plasticity (STDP) rule for self-organizing reservoir computing [12]. Energy efficiency of LSM processors has also been examined via runtime programmable arithmetic precision and data-dependent reconfiguration [10], runtime task-dependent reconfiguration of the reservoir and readout synapse sparsification [12].

The main objective of this paper is to further improve the energy efficiency of LSM neural processors at runtime by

exploring: **1)** sparsity of firing activities in the recurrent reservoir via event-driven processing, and **2)** fine-grained activity-dependent clock gating.

The first proposed technique is motivated by the fact that while the recurrent reservoir is complex, as part of the overall bio-inspired computation model, it inherently facilitates the sparse firing activities, i.e. only a small percentage of reservoir neurons fire at a given time. However, the specific sparse firing structures are application dependent and are not known *a priori*. As such, we explore it by the event-driven based processing for updating membrane potentials of reservoir neurons during runtime. This leads to reduced switching activities and hence boosts energy efficiency. An important point to note is that the benefit of event-driven execution is further amplified by sparse and self-organizing reservoir tuning algorithms such as the one in [12], which we adopt. The on-chip reservoir tunability drives a large number of synaptic weights to zero during the training phase, therefore further reduces synaptic and neural activities.

The second proposed technique addresses the *memory intensive* nature of neural computation. Neural processors including ones that are under consideration require a large amount of storage for synaptic weights and internal neural states. These memory elements heavily load the clock distribution network whose switching activities burn a significant portion of the total power. For this, we recognize that the proposed LSM processor architecture consists of highly regular fabrics of arrays of neural and synaptic elements, providing well-defined boundaries within which storage elements reside. This allows us to partition the on-chip storage into functionally-dependent groups and activate a group only when its associated functions are being processed. The regularity of our LSM processor makes it possible to efficiently partition the individual neuron, leading to fine-grained activity-based clock gating at the granularity of memory elements inside each neuron.

While the presented techniques are generally applicable to the implementation of LSM processors in digital CMOS, we implement them on FPGAs for demonstration purposes. Using the spoken English letters adopted from the TI46 [1] speech corpus as a benchmark, we show that the proposed approaches reduce the training and inference energy consumption of the LSM processor by up to 29% on a Xilinx Virtex-6 FPGA.

II. Background

We briefly introduce the LSM processor design from [12], on top of which we explore the sparsity of firing activities and fine-grained clock gating proposed in this paper.

A. Overall LSM Processor Architecture

The reservoir and the readout layer of Fig. 1 are realized by a reservoir unit (RU) and a training unit (TU), respectively, in the LSM architecture of Fig. 2. Each liquid (reservoir) neuron is implemented with a liquid element (LE) and readout (output) neuron with an output element (OE). External input spikes are sent to their targeted LEs through a crossbar interface. The spikes generated by LEs are registered and sent

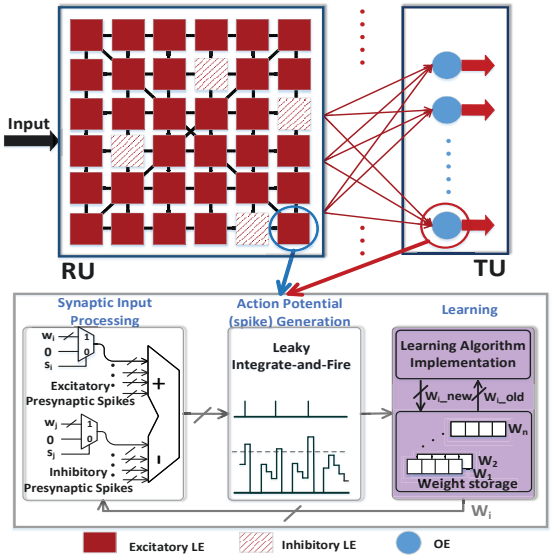

Fig. 2. Hardware implementation of the LSM.

to the TU. Meanwhile, these spikes are also fed back to other LEs in the RU through another crossbar interface.

On-chip learning is executed in two separate phases. First, during the reservoir training phase, the RU is trained by a hardware-friendly spiking-timing dependent plasticity (STDP) algorithm [12] until its synaptic weight distribution converges. Then, a bio-plausible supervised spike-based algorithm [5] is invoked to train the TU, which is responsible for the main classification function. In the second phase, the RU maintains its synaptic weights while producing inputs to the TU.

As in Fig. 2, each LE or OE contains three modules: a synaptic input processing module that processes the second-order synaptic responses from any presynaptic neuron, an action potential (spike) generation module that updates the membrane potential and generates spikes based on the widely used leaky integrate-and-fire (LIF) model, and a learning module that tunes the afferent presynaptic weights of the associated neuron. LEs and OEs differ in terms of the learning rule, arithmetic resolutions employed for digital realization and weight storage. A block memory (BRAM on FPGAs) is used inside each OE to store all its presynaptic weights. LEs, on the other hand, make use of registers because of the lower synaptic bit resolution and the sparser connection in the reservoir.

B. Spike-Timing Dependent Plasticity Reservoir Tuning

The LSM processor targeted in this paper integrates on-chip STDP-based reservoir tuning. STDP is an unsupervised Hebbian learning mechanism realizing synaptic plasticity based on the relative timing of pre- and postsynaptic spike pairs [13]:

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

where Δw^{+} and Δw^{-} represent the weight modification induced by long-term potentiation (LTP) and long-term depression (LTD), and $A_{\pm}(w)$ determines the strength of LTP/LTD.

A straightforward hardware realization of STDP in high bit resolution produces good learning performance but at the cost of large area/power overhead. On the other hand, simply employing low bit resolution in representing synaptic weights and realizing the STDP learning curve leads to an immediate performance loss. To address this challenge, we adopt the hardware-optimized STDP realization with low bit resolution based on a data-driven approach [12]. This leads to a look-up table based implementation with the minimal aggregated discretization error and simple logic.

III. Proposed Event-driven Processing

We propose an event-driven approach for processing the synaptic activities in the recurrent portion of the LSM, i.e. the reservoir. This is based on the observation that the firing activities in the reservoir are typically sparse. That is, at a given moment, only a fraction of the liquid neurons fire, which in turn activates only a subset of the liquid neurons at the subsequent time point. This observed sparsity can be jointly attributed to the basic properties of the LSM as a brain-inspired model and typical sparse encoding used for spike inputs [14].

To see how such sparsity can be explored for advantage in energy efficiency, we recall that the leaky integrate-and-fire (LIF) model can be discretized in time to keep track of the membrane potential of a spiking neuron:

$$V_{mem}(n+1) = V_{mem}(n) - \frac{V_{mem}(n)}{\tau_m} + \sum_{i=1}^{n} R_i, \qquad (2)$$

where $V_{mem}(n+1)$ ($V_{mem}(n)$) is the membrane potential at the $(n+1)$th (nth) biological time step, τ_m is the time constant of the cell membrane, and R_i is the second-order response due to the i-th presynaptic input as described in [5].

Since each presynaptic response is transient and the membrane potential decays exponentially when no synaptic input arrives, conceptually, it is only necessary to check for possible generation of action potential after updating the membrane potential upon receiving at least one input spike. Exploring this sparsity, we opportunistically skip the updating process of membrane potential during the biological time steps when no synaptic input is presented to reduce energy dissipation. Assuming that the number of time steps that are skipped is M, we update the membrane potential in an event-driven fashion according to:

$$V_{mem}(n+M) = V_{mem}(n) - M \cdot \frac{V_{mem}(n)}{\tau_m} + \sum_{i=1}^{n} R_i. \qquad (3)$$

To implement the event-driven processing in our processor, we monitor the arrival of presynaptic spikes of each liquid neuron (LE) using a bitwise OR gate at each time step. If there is no presynaptic spike at the present time step, a skipping-time counter used to track M in (3) would increment by 1 and membrane potential update is skipped. At any subsequent time step when at least one presynaptic spike arrives, the membrane potential update process is triggered according to (3) and the counter is reset to 1. We configure the DSP multiplier IP available on the Xilinx Virtex-6 FPGA to perform 8-bit multiplications according to (3) within one clock cycle while minimizing power dissipation. The same event-driven approach can be applied to the readout neuron (OE). However, due to the full connectivity between the reservoir and the output layer, there is little opportunity for an OE to receive no synaptic input at each time step. As a result, event-driven execution is not pursued for OEs.

IV. Activity-dependent Clock Gating

Neural processors including ones that are under consideration typically require a large amount of storage for synaptic weights and internal neural states. From a power point of view, these memory elements heavily load the clock distribution network and their clock-induced switching activities can lead to a significant amount of power dissipation. We propose a fine-grained activity dependent clock gating strategy to address the power dissipation incurred by the *memory intensive* nature of the targeted LSM processors.

A. Opportunities for Activity-dependent Clock Gating

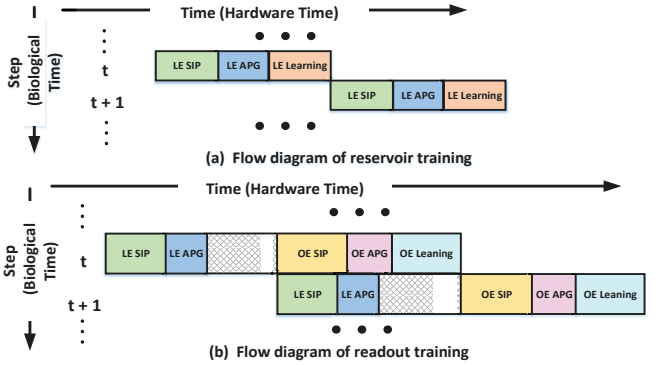

Fig. 3. Flow diagram of reservoir and readout training. (SIP: Synaptic Input Processing, APG: Action Potential (Spike) Generation)

The presented LSM processor architecture consists of highly regular fabrics of arrays of neural and synaptic elements and its operations span across several well-defined steps controlled by the corresponding states of the processor-level global finite state machine (FSM). In our design, there exist global FSMs inside the RU and the TU respectively to parallelize the processing steps of all neurons. Corresponding to the three modules inside a neuron element as shown in Fig. 2, the processing flow for each neuron can be divided into three stages, during each of which only the corresponding module is active. The flow diagram of reservoir and readout training is shown in Fig. 3.

To process each of these steps for a given neuron, a number of memory elements are needed to store the membrane voltage, synaptic weights, and internal neural states. On the FPGA platform, for example, we extensively use registers and block RAMs (BRAMs), latter of which store the weights of the readout synapses. With a global clock driving all registers and BRAMs through a dedicated clock tree on the FPGA, more than 60% of the total processor dynamic power would be dissipated by the clock tree and toggling of the registers and BRAMs.

TABLE I
NUMBERS OF FSM STATES, MEMORY ELEMENT BITS AND CYCLE OCCUPANCIES INSIDE NEURONS.

Neuron Type	# of States	# of Memory Bits	Stage	Clock Cycles	Active Bits
LE	10	87	Synaptic Input Processing	49	40
			Action Potential Generation	3	11
			Learning	32	36
OE	10	1,166	Synaptic Input Processing	271	64
			Action Potential Generation	2	13
			Learning	405	1,089

To this end, we recognize that the regularity of the LSM processor architecture and processing flows provides well-defined boundaries within which storage elements reside. As shown in Table I, the three processing stages consume a varying number of clock cycles and involve different subsets of the registers and BRAMs. This fact allows us to partition the on-chip storage for each neuron into different groups, one for each processing stage, leading to a fine-grained activity-based clocking at the granularity of memory elements inside each neuron.

B. Fine-grained Activity-dependent Clock Gating Realization

Since in this work we chose the FPGA as our demonstration platform, for the sake of discussion, we first describe the realization of the proposed fine-grained activity-dependent clock gating on the FPGA. Later, we will also discuss additional opportunities for ASIC-based implementations.

The mapped clock distribution for the LSM processor design is illustrated in Fig 4, where the leaf nodes of the clock tree drive the storage elements inside each neuron. On FPGAs, dedicated routing resources are typically used to ensure the low-skew clock signal delivery across a fairly large design. Under this circumstance, directly gating the clock signal is not a good practice since it involves the use of unconstrained reconfigurable logic and incurs additional clock signal routing. As a result, doing so jeopardizes the low-skew performance ensured by dedicated resources for clock distribution.

With above FPGA design constraints in mind, instead, we lower the power dissipation by utilizing the clock enable signal to reduce the clock-triggered switching activities within the memory elements. In the proposed activity-dependent clock enabling scheme, in each neuron, the memory elements of the same module, which corresponds to a particular processing stage, share a common clock enable signal. If the memory elements are implemented as registers, this clock enable signal will be connected to the corresponding local CE inputs of each slice, which is the basic building block of an FPGA. For the BRAM inside each OE, the clock enable signal is directly enable or disable the memory clock input. For both the RU and the TU, the current state of the global FSM is encoded to produce the clock enable signal for each module using combinational logic as depicted in Fig 5. When the neural processor operates in a given stage, only the corresponding clock enable signal would be asserted to activate memory elements inside this module while the memories associated with other modules are deactivated to save power.

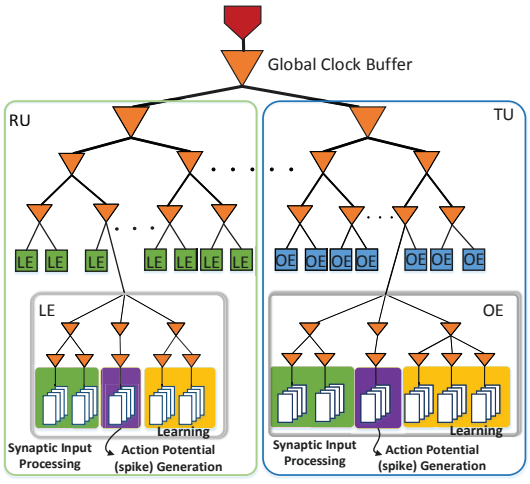

Fig. 4. Clock distribution of the LSM processor.

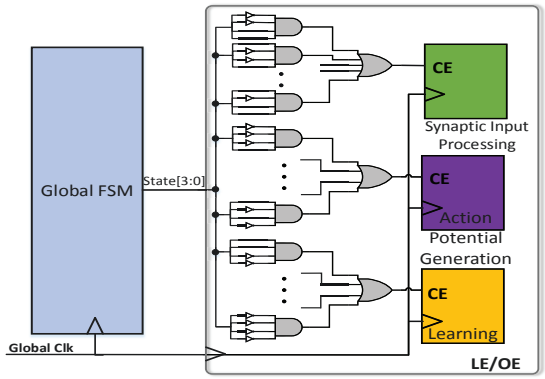

Fig. 5. The implementation of clock enabling.

Although clock enabling/disabling each module independently of other two modules takes the maximal clock gating opportunities across different processing stages, doing so incurs maximal extra logic and power overhead for the implementation. As one example, our experimental data has shown that having three independent clock enabling signals in the LE does not significantly reduce the power of the RU since the power overhead incurred for implementing clock enabling largely offsets its benefit.

By analyzing Table I, we note that the action potential generation stage of the LE only takes 3.5% clock cycles in each biological time step during the reservoir training phase. Moreover, only a small percentage of registers are accessed during this stage. These observations suggest that we may use a shared clock enable signal for both the synaptic input processing and the action potential generation module. Alternatively, we do not choose to combine the clock enable signals of the action potential generation module with the learning

module. This is because during the readout training and the final classification phase, the learning module can be fully disabled to get a large energy benefit. During the two phases, however, the action potential generation module must be active for updating the membrane voltage. The same observation can be made for OEs. Therefore, similarly, a shared clock enable signal is implemented to control the synaptic input processing and the action potential generation module in an OE.

The partition of the on-chip storage inside each neuron is based on the unique characteristics of the proposed LSM processor architecture and largely independent of the specific implementation platform. Therefore, the proposed activity-dependent clock gating approach can be implemented to an LSM processor in general and similar benefits would be expected across different platforms. Moreover, the above clock enabling approach does not reduce the power dissipated by the clock tree itself on the FPGA platform. Since ASIC implementations are not restricted by the aforementioned FPGA clock routing constraints, direct clock gating may be added on top of the proposed activity-dependent clock enabling approach to further reduce power dissipation.

V. Experimental Results

The proposed LSM neural processor is built on a Xilinx Virtex-6 FPGA platform with 135 reservoir neurons and 26 readout neurons and simulated for the application of speech recognition. Activity-based power analysis is carried out to measure the energy dissipation. The adopted speech benchmark consists of a subset samples from the TI46 corpus [1], which includes 260 speech samples of 10 utterances of each English letter from "A" to "Z" recorded from a single speaker. Each speech sample is pre-processed into 78 channels of spike trains with approximate 680 time steps in length on average. The proposed neural processor can achieve a 93.1% classification performance with the adopted benchmarks.

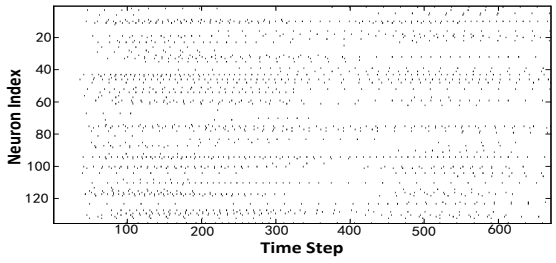

Fig. 6. The reservoir spike raster with the letter "H" presented as the input.

To demonstrate the sparsity of firing activities that we explore, in Fig. 6, we show a typical spike raster plot for the reservoir after reservoir training when the letter "H" is presented as the input. The spike raster has a high sparsity level of 1.75%.

Fig 7 illustrates the sparsification of reservoir connectivity introduced by the self-organizing reservoir tuning algorithm [12]. Fig. 7(a) plots the initial randomly generated reservoir synaptic weights. The dark-gray vertical bar of each neuron stacks multiple squares, each representing a synapse with non-zero initial weight. The height of the bar indicates

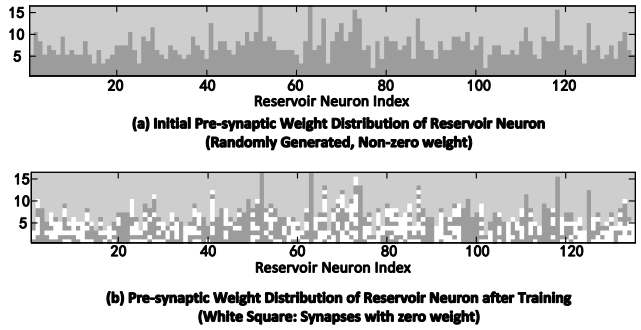

Fig. 7. Sparser reservoir connectivity introduced by the self-organizing reservoir tuning.

the number of presynaptic connections, which is set to 16 at a maximum per neuron. The application of reservoir tuning zeros out about 42% of synapses, indicated by white squares in Fig. 7(b), which further reduces the activity level in the reservoir.

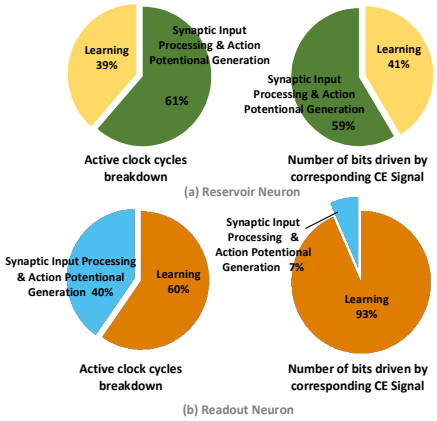

Fig. 8. The portion of each enable clock signal contributes in the reservoir and the readout neuron in terms of active time and number of load bits.

To shed some light on the potential benefit brought by the proposed activity-based clock gating, we report the percentage of clock cycles during which each processing stage is activated and the percentage of the memory used in each stage. The data are shown in Fig. 8 for training of liquid and output neurons, respectively. As expected from the figure, it is clear that the storage elements and the processing of liquid neurons are split between two relatively balanced major components: the combined synaptic input processing and action potential generation module and the learning module. For output neurons, the learning module is dominant in terms of storage and contributes to the major part of the neuron processing time. Hence, the proposed clock gating technique is expected to noticeably reduce energy dissipation.

For comparison purposes, we use a "baseline" LSM processor which does not incorporate the two proposed energy reduction techniques as a reference. First, we compare the resource utilization of the baseline with the proposed LSM in terms of slice flip-flops (FFs) and slice LUTs as well as their percentages of usage with respect to the available resources of the FPGA board in Table II. It is evident that implementing the proposed techniques incurs low hardware overhead.

TABLE II
COMPARISON ON HARDWARE RESOURCE UTILIZATIONS.

	FFs (% of total)	LUTs (% of total)	Normalized FFs	Normalized LUTs
Baseline	14510 (4.8%)	56808 (37.8%)	1.00	1.00
Proposed	14914 (4.9%)	58962 (39.1%)	1.03	1.04

We are aware that the Xilinx design tools offer a standard intelligent clock gating in general [15] by preventing logic not used in a given clock cycle from toggling. To better illustrate the energy efficiency of the proposed design, we have another reference when comparing the power consumption, which is called the "tool gating" LSM that implements the standard clock gating provided by the Xilinx ISE on top of the baseline design. Based on FPGA activity-based simulation data, the dynamic powers of the three LSM processors clocked at 100MHz are given in Table III.

TABLE III
COMPARISON ON AVERAGE POWER DISSIPATIONS.

	Reservoir Training Power (W)	Readout Training Power (W)	Classification Power (W)
Baseline	0.472	0.389	0.398
Tool Gating	0.400	0.367	0.373
Proposed	0.320	0.279	0.281

Using the power data from Table III, we compare the three designs in terms of the energies consumed for training and classifying a representative speech sample in Table IV. To get good learning performance, here 25 epochs of reservoir training and 250 epochs of readout training are conducted for the speech sample. The training energy accounts for the energies consumed for both reservoir and readout training.

TABLE IV
COMPARISON ON THE ENERGIES CONSUMED FOR TRAINING AND CLASSIFYING A SPEECH SAMPLE.

	Training (mJ)	Classify (mJ)	Normalized Training	Normalized Classify
Baseline	452	0.87	1.00	1.00
Tool Gating	426	0.82	0.94	0.94
Proposed	324	0.62	0.72	0.71

The results show that the proposed event-driven processing and fine-grained clock gating can give rise to a considerable amount of energy reduction. As a result, the proposed recurrent spiking neural processor is 28% more energy efficient for training and 29% more energy efficient for inference than the baseline while incurring little extra hardware resource. It is also demonstrated that the proposed LSM processor is more energy efficient than the tool-gated design with standard clock gating. It is reported that the clock gating implemented by the tool only apply clock enable signals to the weight storage elements (i.e. weight registers in LEs and BRAMs in OEs), which may indicate that the unique regularities of the LSM architecture are not fully aware. In comparison, the proposed clock gating method takes full advantage of the unique architectural and functional properties of the LSM processor and implement fine-grained clock enable signals on all storage elements in each neuron.

VI. CONCLUSIONS AND FUTURE WORKS

We improve the energy efficiency of the LSM based recurrent spiking neural processors by proposing event-driven processing and fine-grained activity-dependent clock gating. These two techniques naturally explore the architectural properties and runtime characteristic of the LSM architecture. The efficacy of the proposed approaches has been verified via an FPGA prototype which leads to significant energy reduction for speech recognition.

As one of the most profound real-world applications used in most reservoir computing works, the speech recognition is chosen to benchmark our LSM processor for the purpose of demonstrating the effectiveness of the proposed event-driven processing and fine-grained activity-dependent clock gating approaches. In the future, various other applications with large size dataset will be examined on the LSM processor to further illustrate its energy efficiency.

REFERENCES

[1] "The TI46 Speech Corpus," http://catalog.ldc.upenn.edu/LDC93S9.

[2] M. LukošEvičIus and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[3] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, "Isolated word recognition with the liquid state machine: a case study," *Information Processing Letters*, vol. 95, no. 6, pp. 521–528, 2005.

[4] A. Ghani, T. M. McGinnity, L. P. Maguire, and J. Harkin, "Neuro-inspired speech recognition with recurrent spiking neurons," in *Artificial Neural Networks-ICANN 2008*. Springer, 2008, pp. 513–522.

[5] Y. Zhang, P. Li, Y. Jin, and Y. Choe, "A digital liquid state machine with biologically inspired learning and its application to speech recognition," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 11, pp. 2635–2649, 2015.

[6] S. Mitra, S. Fusi, and G. Indiveri, "Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, no. 1, pp. 32–42, 2009.

[7] T. Schoenauer, S. Atasoy, N. Mehrtash, and H. Klar, "Neuropipe-chip: A digital neuro-processor for spiking neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 205–213, 2002.

[8] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "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*, vol. 34, no. 10, pp. 1537–1557, 2015.

[9] Q. Wang, Y. Jin, and P. Li, "General-purpose LSM learning processor architecture and theoretically guided design space exploration," in *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*. IEEE, 2015, pp. 1–4.

[10] Q. Wang, Y. Li, and P. Li, "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, 2016, pp. 361–364.

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

[12] Y. Jin, Y. Liu, and P. Li, "SSO-LSM: A sparse and self-organizing architecture for liquid state machine based neural processors."

[13] G.-q. Bi and M.-m. Poo, "Synaptic modification by correlated activity: Hebb's postulate revisited," *Annual review of neuroscience*, vol. 24, no. 1, pp. 139–166, 2001.

[14] B. Schrauwen and J. Van Campenhout, "BSA, a fast and accurate spike train encoding scheme," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, 2003, pp. 2825–2830.

[15] "Xilinx Intelligent Clock Gating," https://www.xilinx.com/support/documentation/application_notes/xapp790-7-series-clock-gating.pdf.