System Design for In-hardware STDP Learning and Spiking Based Probablistic Inference

Khadeer Ahmed, Amar Shrestha, Yanzhi Wang, Qinru Qiu
Department of Electrical Engineering and Computer Science, Syracuse University, NY 13244, USA
Email {khahmed, amshrest, ywang393, qiqiu} @syr.edu

***Abstract*— The emerging field of neuromorphic computing is offering a possible pathway for approaching the brain's computing performance and energy efficiency for cognitive applications such as pattern recognition, speech understanding, natural language processing etc. In spiking neural networks (SNNs), information is encoded as sparsely distributed spike trains, enabling learning through the spike-timing dependent plasticity (STDP) mechanism. SNNs can potentially achieve ultra-low power consumption and distributed learning due to the inherent asynchronous and sparse inter-neuron communications. Several inroads have been made in SNN implementations; however, there is still a lack of computational models that lead to hardware implementation of large scale SNN with STDP capabilities. In this work, we present a set of neuron models and neuron circuit motifs that form SNNs capable of in-hardware fully-distributed STDP learning and spiking based probabilistic inference. Functions such as efficient Bayesian inference and unsupervised Hebbian learning are demonstrated on the proposed SNN system design. A highly scalable and flexible digital hardware implementation of the neuron model is also presented. Experimental results on two different applications: unsupervised feature extraction and inference based sentence construction, have demonstrated the proposed design's effectiveness in learning and inference.**

Keywords—STDP Learning; bayesian neuron; unsupervised feature learning; bayesian inference; Spiking neural network; winner-take-all; digital neuron

I. INTRODUCTION

The exponential growth of data over the past decade has generated a need for higher processing capability, low energy consumption and high scalability. Limitation of the Von Neumann architecture and barriers such as memory capacity, power density etc. in the CMOS technology are being highly tested to meet today's requirements and also to fulfill Moore's predictions. These limitations have motivated novel research efforts in bio-inspired computing, which imitates the structure and function of the brain, the computing engine that is able to process massive amounts of real-time information with less than 20 Watts of power consumption [1].

The processing capability of brain comes from the collective processing abilities of simple processing components i.e., neurons. Interconnected neurons form the basis of a neural network. The ability of neural networks to perform pattern recognition, classification and associative memory, is essential to applications such as character recognition, speech recognition, sensor networks, decision making etc. [2] [3] [4] [5] [6]. Spiking neural networks (SNNs), which use spikes as the basis for communication, is the third generation of neural networks inspired by the biological neuron models [7].

The SNN has the potential to reach very low energy dissipation since each neuron works asynchronously in an event-driven manner. Moreover, fully distributed Spike Timing Dependent Plasticity (STDP) learning [8] can be achieved on SNNs, which updates synaptic weight based only on local information of individual neuron. The emerging stochastic SNN that generates spikes as a stochastic process is not only more biologically plausible [9] but also enhances unsupervised learning and decision making [10] [11]. It further increases the fault tolerance and noise (delay) resilience of the SNN system since the results no longer depend on the information carried by individual spikes but the statistics of a group of spikes.

The potential benefits of the SNN cannot be fully realized without dedicated hardware because full software implementations have high coordination overheads and are limited by the allowable degree of parallelism. However, many traditional computational models of SNN are not designed to facilitate hardware implementation [12] [13]. They either consist of excessive physiological details [14] [15] or rely on centralized control to coordinate neurons [16]. Some recent research works on SNNs have been carried out from the hardware design perspective [5] [17] [18]. Novel hardware systems such as IBM's TrueNorth neurosynaptic processor has enabled breakthrough in design and applications of SNN. However, STDP learning has not been an integral part of the neuron model in these hardware systems. As a result, they do not support real-time in-hardware learning, which is critical when being applied to a dynamic environment or multiple applications. Although hardware implementation of STDP learning has been discussed in several previous publications [19] [20], these works focus more on circuit and device level analysis on how variable synaptic plasticity is achieved. They either did not demonstrate the ability of learning [20] or were applied only to small scale problems with linearly separable classes [19]. Furthermore, these implementations are either in analog domain or rely on certain non-linear properties of the device while no specific computational model was provided.

In this work we focus on a large scale digital hardware implementation of the stochastic SNN with biologically plausible in-hardware learning. An improved computational model of the stochastic SNN is presented. The model describes neuron behavior, STDP learning rules and network topology. A reference digital implementation of the neuron model is also provided, which is highly scalable and flexible. Our main contributions are summarized as follows.

1. This is the first work that utilizes Bayesian and ReLU neurons to form motifs of neuron circuit for learning and inference. The Bayesian neurons integrate and fire stochastically to generate excitatory spikes, while the ReLU neuron performs temporal coding to convert input excitation to a sequence of inhibitory spikes. Connected Bayesian and ReLU neurons realize the winner-take-all function. This not only improves the quality of learning by enhancing the selectivity and specificity, but also helps to normalize the spiking activities during the inference process.
2. By introducing the ReLU neuron, the inhibitory input of each neuron is now spiking based instead of amplitude based. Therefore, the excitatory and inhibitory synaptic models in our SNN are unified. Compared to previous models that use spiking based excitatory inputs and amplitude based inhibitory inputs [16], our model is simpler for hardware implementation. Furthermore, spreading a large inhibition signal to a sequence of spikes helps to keep the neurons in their dynamic range.
3. The stochastic firing and winner-take-all functions improve the STDP learning quality. When applied to unsupervised feature learning of the MNIST dataset in a convolutional setting, the proposed stochastic SNN learns the features that can be classified with more than 94% accuracy, which is 3% higher than [4] and close to the best result in [21], by using 23% less excitatory neurons and much

This work is partially supported by the National Science Foundation under Grants CCF-1337300.

simpler computations.

4. A reference hardware implementation of the generic neuron model is presented. With careful resource sharing and pipelining, each hardware neuron requires only two adders, one multiplier and 209KB of memory to instrument both learning and inference for up to 256 logical neurons with 16-bit data precision. The proposed design achieves a maximum spiking rate at 2000Hz (which is about 10 times of the average firing rate of neurons in an active brain.) Tradeoffs between hardware cost and classification accuracy is discussed.
5. Comprehensive experimental results on two different applications: unsupervised feature extraction and inference-based sentence construction, have demonstrated the effectiveness of the proposed stochastic SNN in terms of in-hardware learning and inference.

II. Related Work

A dedicated hardware implementation of the SNN is a very attractive option for a large variety of applications due to its significant potential in energy efficiency. The biological neuron models are bulky and complicated, thus not suitable for large-scale implementations. Neurogrid, developed at Stanford University for simulation of biological brains [22], uses analog circuits to emulate the ion channel activity and uses digital logic for spike communication. BrainScaleS is another hardware implementation that utilizes analog neuron models to emulate biological behavior [23]. These implementations have been focusing on biologically realistic neuron models and are not optimized for large-scale computation. IBM has come up with the TrueNorth architecture which is digital and optimized for large-scale applications and contains 4096 cores with 256 neurons in each core [24]. None of the above hardware systems support real-time in hardware STDP learning. Several existing efforts address hardware implementation of the STDP function [19] [20]. Their main focus is how to use nonlinear property of resistive RAM [19] or analog circuit [20] to realize variable synaptic weights. [19] applies a ReRAM array to memorize the EEG signal of three vowels and [20] does not provide experimental data to demonstrate the circuit's ability to learn. Neither of them give details of their computational model.

Models of spiking neurons have been studied in previous works. Authors of [2] introduced a model that uses active dendrites and dynamic synapses with an integrate-and-fire neuron for character recognition. Reference [3] implements spiking self-organizing maps using the leaky-integrate-and-fire neurons for phoneme classification. Work in [5] uses the Siegert approximation for integrate-and-fire neurons to map an offline trained deep belief network onto an event-driven SNN suitable for hardware implementation. Work in [6] implements a large-scale model of a hierarchical SNN that integrates a low-level memory encoding mechanism with a higher-level decision process to perform visual classification task in real-time. It models Izhikevich neurons with conductance-based synapses and uses STDP for memory encoding.

All of the above mentioned models have deterministic functions. The stochastic nature in spike patterns has already been found in the lateral geniculate nucleus (LGN) and the primary visual cortex (V1) [9]. Ignoring the randomness in a neuron model not only limits its effectiveness in sampling and probabilistic inference related applications [14] [15], but also reduces its resilience and robustness. A stochastic neuron model and its STDP learning rule are presented in [16]. The resulting network is used to classify simple patterns. We improve this model for a superior learning ability and a simpler hardware implementation. We also apply the network for large scale applications such as unsupervised feature extraction for the MNIST dataset.

III. Improved Model for Stochastic SNN

A. Existing model and limitations

Fig. 1 shows a stochastic neuron model proposed in [16]. The membrane potential $u(t)$ of neuron Z is computed as the difference between the excitatory input and the common inhibition $I(t)$.

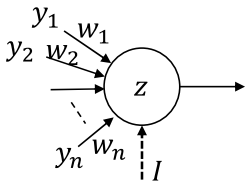


Fig. 1. A generic neuron model with centralized inhibition

$$u(t) = w_0 + \sum_{i=1}^{n} w_i \cdot y_i(t) - I(t) \quad (1)$$

where w_i is the weight of the synapse connecting Z to its i^{th} presynaptic neuron y_i, $y_i(t)$ is 1 if y_i issues a spike at time t (otherwise is 0), and w_0 models the intrinsic excitability of the neuron Z. The stochastic firing model for Z, in which the firing probability depends exponentially on the membrane potential, is expressed as

$$prob(Z\ fires\ at\ time\ t) \propto \exp(u(t)) \quad (2)$$

The learning function involves updating the weight w_i of i^{th} synapse and the intrinsic excitation w_0 of the neuron. Their changes are calculated as:

$$\Delta w_i = \begin{cases} ce^{-w_i} - 1, & if\ spike\ occured\ in\ STDP\ window \\ -1, & if\ there\ is\ no\ spike\ in\ STDP\ window \end{cases} \quad (3)$$

$$\Delta w_0 = e^{-w_0} \cdot z - 1 \quad (4)$$

The above equations are scaled with a constant learning rate to perform the final update. It can be proven that [16], based on this learning rule w_i converges to the log of the probability that the presynaptic neuron y_i fired within the STDP window before neuron Z fires, and the firing probability of Z calculated by Eqn. (2) is the Bayesian probability of Z given the condition of its input neurons $y_1, y_2, \ldots\ y_n$.

This neuron model has a dedicated common inhibition signal $I(t)$ for every neuron. The inhibition is generated by a central controller, which limits the scalability of the SNN system. Moreover, the inhibitory synapse (which provides $I(t)$ signal) is amplitude (instead of spiking) based, which results in non-uniformity in synapses and significantly increases the complexity for hardware implementations. Obviously, the generation and utilization of amplitude based inhibition is not compatible with the rest of the spiking based system. Also, this generic model is difficult to stabilize. A small change in the membrane potential will be amplified due to its exponential relation with the firing probability.

To overcome these shortcomings, we propose to improve the original model by: (1) using ReLU neurons to convert a large inhibition signal to a sequence of spikes, (2) unifying the excitatory and inhibitory synapses, (3) limiting the range of membrane potential $u(t)$ of the Bayesian neurons, (4) connecting the Bayesian neurons and ReLU neurons to form the winner-take-all (WTA) circuit, which is the building block of proposed stochastic SNN. The details of these improvements are discussed below.

B. Improved inhibitory spike generation using ReLU

A Bayesian neuron is memoryless and its firing rate depends on the instantaneous membrane potential, which is limited to a relatively narrow range in order to maintain the stability of the system. Any input outside the range will be truncated. This affects the inhibition process significantly since all the neurons inhibits each other, and the accumulated amplitude of the inhibition spikes will have a large range. Our solution is to spread the large inhibition signal over time. Instead of using amplitude to indicate the strength of inhibition, we spread it over time and use the duration of spikes to represent the strength of inhibition. This conversion mechanism is achieved by using a spiking Rectified Linear Unit (ReLU) neuron.

The ReLU function is defined as $Z = max(0, u(t) - U_{th})$ where Z is either one or zero, U_{th} is a constant threshold, and $u(t)$ is the membrane potential of this neuron calculated as $u(t) = u(t-1) + \sum_{i=1}^{n} w_i \cdot y_i(t) - ZU_{th}$. In other words, the membrane potential

of a ReLU neuron accumulates every weighted input spike and discharges over time as a burst firing pattern. In our implementation, the spiking threshold U_{th} is set to 1, and after each spike generation, the membrane potential is reduced by the threshold value. This assures that accumulated membrane potential is discharged faithfully over time.

Converting the inhibition signal from the amplitude domain to time domain not only helps to keep the neuron in its dynamic range, but also leads to simplified hardware design, as will be shown next.

C. Improved Bayesian neuron model

In our model the excitatory and inhibitory inputs are treated in a uniform way since both are spiking based. Synapses with a positive weight induce excitation and synapses with negative weight provide inhibition. In this way, the dedicated common inhibition $I(t)$ is no longer needed. Eqn. (1) can now be rewritten as:

$$u(t) = w_0 + \sum_{i=1}^{n} w_i \cdot y_i(t) \tag{5}$$

In Eqn.(2), small variations of $u(t)$ resulting from the synaptic weight changes will have an exponential impact on the firing probability, which is not desirable. To mitigate this effect we introduce a *range mapping function*. This function is a parameterized sigmoid function for representing more flexible S-shaped curves:

$$u'(t) = A + B/(1 + \exp(-(u(t) - C) \cdot D)) \tag{6}$$

The above equation has four parameters for shape tuning. Parameters; *A* provides the Y-axis offset, *B* performs scaling along the Y-axis, *C* provides the X-axis offset and finally *D* performs scaling along the X-axis. It maps a range of $u(t)$ to a different range $u'(t)$and the out-of-range $u(t)$ to asymptotic values of the function. In this way we can make sure that the membrane potential always lies within the dynamic range of the neuron. After mapping, $u(t)$ in Eqn. (2) should be replaced by $u'(t)$.

Two examples for Eqn. (6) are shown in Fig. 2. Curve (a) expands the input range (10, 20) to the output range (-10, 50). Any input value outside (10, 20) is asymptotically mapped to -10 or 50. Curve (b) compresses the input range (10, 60) to an output range (-10, 10) with asymptotic values for out-of-range inputs. For specific applications, the network topology is given and so is the possible range of the synaptic weights. Therefore, it is not difficult to have a rough estimation of the maximum and minimum values of the accumulated inputs of a particular neuron and a range mapping function can be chosen accordingly.

To sum up, the proposed neuron model handles all synaptic inputs uniformly, and its membrane potential can be constrained within a fixed range. Such regularity reduces the hardware implementation complexity. The range-limiting function also provides the means to adjust the membrane potential and consequently the firing rate across neurons without retraining the entire network.

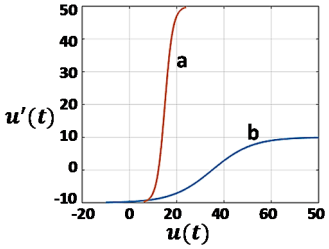


Fig. 2. Range modifier behavior

To obtain Poisson spiking behavior we adopt the method introduced in [25]. The spiking rate $\lambda(t)$ is an exponential function of the inputs, which is represented by Eqn. (7). To generate a Poisson process with time-varying rate $\lambda(t)$, the *Time-Rescaling Theorem* is used. According to the theorem, when spike arrival times v_k follow a Poisson process of instantaneous rate $\lambda(t)$, the time-scaled random variable $\Lambda_k = \int_0^{v_k} \lambda(v)dv$ follows a homogeneous Poisson process with unit rate. Then the inter-arrival time τ_k satisfies the exponential distribution with unit rate.

$$\tau_k = \Lambda_k - \Lambda_{k-1} = \int_{v_{k-1}}^{v_k} \lambda(v)dv \tag{7}$$

To find the next spiking time v_k, we generate a random variable satisfying exponential distribution with unit rate, which represents τ_k. The integral in Eqn. (7) cumulates the instantaneous rates from Eqn. (2) over time until the integral value is greater than or equal to τ_k. Once this happens it implies that the interspike interval has passed and a spike is generated accordingly. Using this method we can generate the Poisson spiking behavior based on the state of the neuron.

D. Winner-take-all circuit

The WTA circuit is a recurrent network where a set of neurons compete with each other for activation. Fig. 3 shows a neural circuit to laterally inhibit a group of Bayesian neurons in a *winner-take-all* manner. Each Bayesian neuron has an associated ReLU neuron, which collects and integrates input from other Bayesian neurons within this competing set and convert the accumulated signal into a sequence of inhibition spikes. The model avoids centralized generation of the inhibition signal, therefore there is no need to synchronize neuron activities. This makes hardware implementation simple and distributed.

Hard or soft WTA behavior can be achieved based on the degree of inhibition delivered. *Hard WTA* happens when the inhibition is strong such that it brings down the firing rate of the non-preferred Bayesian neurons to zero, resulting in only one neuron with the highest excitation being active. Hard WTA can be used for unsupervised feature extraction for enabling each neuron to learn a unique feature. On the other hand, if plural voting action is required within the set, the degree of inhibition is tuned to be moderate. This makes Bayesian neurons fire with different stable rates which is the *soft WTA* behavior where firing rates are proportional to their relative excitation levels. Soft WTA helps to retain more information in probabilistic inference.

The WTA circuit is the basic building block for our SNN. In Section V we demonstrate that it can be used to implement a set of filters for kernel feature extraction in the convolutional layer for image recognition, or a lexicon of words or phrases for language processing.

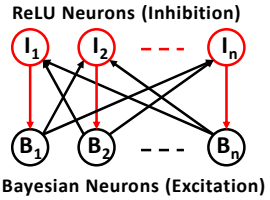


Fig. 3. Winner-take-all circuit

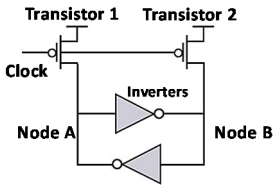


Fig. 4. Bistable 6-T random number generator design.

IV. Hardware Architecture of Digital Neuron Model

Keeping reliability, scalability and flexibility as the primary focus we choose digital implementation over analog. This section presents a reference design of a hardware neuron. Two functions are supported by the hardware, (1) membrane potential update and spike generation, also referred as the "recall" function, (2) synapse weight update based on STDP rule, also referred as the "learning" function. For lower hardware cost and power consumption, each set of hardware is used to evaluate the recall and learning function of multiple neurons in the SNN in a pipelined manner. We refer to the hardware implementation as *physical neurons*, and the neurons in the computational model as *logical neurons*.

A. Major components

Approximation and resource sharing techniques are adopted in the proposed design to reduce hardware complexity. Instead of directly implementing Eqn. (6), we approximate it using a piecewise linear func-

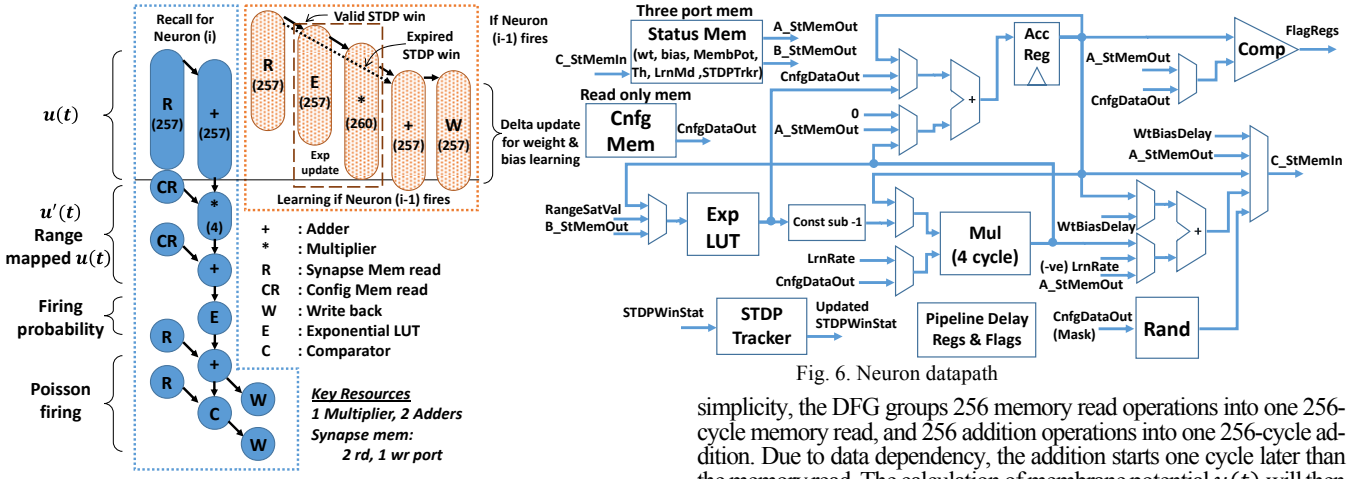


Fig. 5. Dataflow graph for pipelined recall and learning

Fig. 6. Neuron datapath

tion. The asymptotic regions of the curve are approximated with constant values, while the rest of the curve is approximated with a straight line represented as $u'(t) = m \cdot u(t) + c$ where m is the slope of the line and c is the Y-axis intercept. In this way, the range mapping function can be implemented using a multiplier and an adder. The exponential relation between the firing rate and membrane potential (i.e. Eqn. (2)) is realized with an exponential lookup table, which reduces the compute and area requirement.

To generate the Poisson firing pattern as described by Eqn. (7), we use a geometric distribution instead of an exponential distribution as discrete values are required. A random number is drawn whenever a spike is generated. The traditional shift register based pseudo random number generator will require a significant amount of area and power with a limited degree of randomness. Motivated by the Intel's design for Ivy Bridge [26], we adopt a 6-transistor (6-T) random number generator as shown in Fig. 4, which comprises a bistable structure and two pull-up transistors. When the clock signal is in the low phase, both Node A and Node B are pulled up (close) to V_{dd}. When the clock signal is in the high-phase, both pull-up transistors are cut-off and the bistable begins evaluation phase. During evaluation, any noise or disturbance will drive the bistable out of the unstable equilibrium point (Node A = Node B = $V_{dd}/2$) and make one of them logic 1. Nodes A and B will become logic 1 with equal probability because the noise will incur equal probability of positive and negative effects on the node voltage. Extensive experiments have been conducted on this design, demonstrating its effectiveness and 5% tolerance level on process variations.

This random number generator is used to compute the geometric distribution which represents the next spike generation time. The significant values in a geometric distribution lie within a small range, hence the random number is directly used with a programmable mask for fine tuning. The exponential lookup table along with multiplier and an additional adder is used to realize Eqn. (3) and (4).

Two memory banks are needed in the hardware implementation. The first one is a configuration memory, which stores the parameters such as learning rate and the coefficients of the range mapping function. The second is the neuron status memory. It stores the weight and the accumulated inter-spike time (as calculated by Eqn. (7)) for each synapse and STDP window status. Each neuron requires 806 bytes of neuron status memory and 13 bytes of configuration memory.

B. *Dataflow graph and architecture*

Fig. 5 shows the data flow graph (DFG) of the learning and recall functions of a Bayesian neuron. The operations in blue are required for both learning and recall functions. They calculate the membrane potential and evaluate spike firing conditions. The weight of each input synapse is read from status memory and accumulated according to Eqn. (5). We assume that each neuron has a maximum of 256 input synapses. For simplicity, the DFG groups 256 memory read operations into one 256-cycle memory read, and 256 addition operations into one 256-cycle addition. Due to data dependency, the addition starts one cycle later than the memory read. The calculation of membrane potential $u(t)$ will then go through range mapping, spiking rate generation and Poisson firing steps as indicated in the DFG.

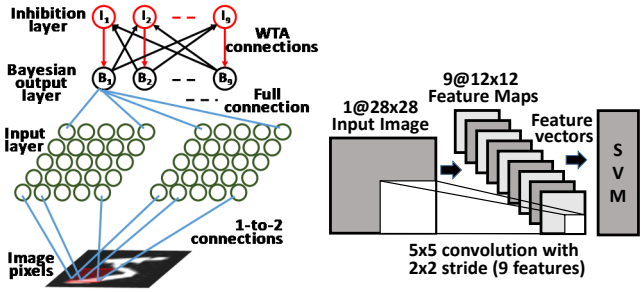
The operations in orange are required only for the learning function. They will be executed only when learning is enabled and either a spike is issued within the STDP window or the STDP window has expired. In both cases, the original synaptic weight will be read out and the updated synaptic weight will be written back. If a spike is issued within the STDP window, the exponential LUT lookup and multiplication will be executed to calculate the Δw_i according to the first part of Eqn. (3). Otherwise, if the STDP window expires, the *Exp update* block will be skipped and a constant negative Δw_i will be used to update the synaptic weight according to the second part of Eqn. (3). No action will be taken in all other cases. Our simulation shows that these learning related operations are only executed with less than 17% probability during the learning stage. The fact that online learning is performed much less frequently compared to recall leads to lower total power consumption. Again, the operations performed on 256 synaptic weights (w_i) plus one intrinsic weight (w_o) is chained into a 257-cycle operation. To achieve high throughput, a 4-stage pipelined multiplier is used. As a result it takes 260 cycles to process 257 multiplications in the data flow graph. Because the synaptic weight can only be updated after the condition for firing is evaluated, the learning function of the (i-1)th logical neuron can overlap with the recall function of the i^{th} logical neuron.

An analysis on the data flow graph shows that two adders and one multiplier are needed as computational resources. It also shows that the neuron status memory must have two read ports and one write port. With these resources the overall latency to evaluate the recall and learning functions of a logical neuron is 526 cycles and the throughput is 267 cycles per logical neuron. We define the time to evaluate all 256 neurons in a core as the *neuron evaluation cycle* (*NEC*). One NEC consists of 267*256 + 259 = 68,611 clock cycles.

Based on the above analysis, we developed a digital architecture with 16-bit fixed-point precision for the neuron model encompassing both recall and learning circuits. Fig. 6 shows the datapath of our design. The controller is divided into two state machines, one for recall and the other for learning. By disabling the learning module and stochastic firing function, the same design can be used to implement integrate and fire as well as the ReLU neurons.

V. Experiments

To validate its functionality, we applied the neuron model on two different applications. In the first experiment, the neuron model is used to perform unsupervised feature learning and extraction of hand written digits. With this experiment, we will demonstrate the in-hardware learning capability of the neuron model. The potential tradeoff between hardware complexity using fixed point arithmetic and detection quality will be discussed. The second experiment demonstrates the model's capability of performing Bayesian inference, where it is applied for



(a) (b)

Fig. 7. Network structure for (a) training and (b) testing

sentence construction with a learned natural language model.

A. *Unsupervised Feature Learning and Extraction*

The stochastic firing and STDP learning enables unsupervised feature learning and extraction, which is the function of the base layer in every convolutional network for image recognition. The MNIST dataset is used for the proposed model to learn features of handwritten digits ranging from '0' to '9'. We use 2000 randomly selected samples from the training set to learn the features and tested against 2000 images randomly picked from the testing set. For all the experiments we use binary MNIST images.

A convolutional neural network was constructed using the Bayesian neurons. Kernels with two different sizes are tested, 5x5 and 7x7. For both kernel sizes we set the X and Y strides to be 2 pixels. Each kernel is mapped to 9 features, implemented by 9 Bayesian neurons in the output layer. Each neuron of the Bayesian output layer is connected to all input neurons of the kernel. The input neurons perform population coding of input pixels, with two neurons representing black and white value of each input pixel. The neurons in the input layer fire, facilitating the Bayesian neurons to fire. Based on their relative spike-timings, the weight of the synapse is updated. A ReLU neuron based inhibition layer is attached to the output layer and implements the hard WTA function to ensure that only one feature will be activated for each kernel and each Bayesian neuron learns a unique feature. The setup for learning and testing a 5x5 kernel is shown in Fig. 7. The input layer consist of 50 neurons, and the output and inhibition layers both have 9 neurons. When an input neuron is active, it fires at a 10% probability. The learning rate is fixed at 0.01, and the STDP period is 30 neuron evaluation cycles for the experiments.

The stochastic SNN performs learning and feature extraction functions similar to a Convolutional Restricted Boltzmann Machine (CRBM) [27]. The same set of training and testing images is applied to an open source software implementation of CRBM. We found that they produce comparable feature maps and filtered images as shown in Fig. 8. A support vector machine (SVM) classifier is used to check the effectiveness of the learnt features. Two different SVMs are trained and tested using features extracted from our stochastic SNN and CRBM. The results show that the features extracted by stochastic SNN and CRBM can be used to classify with 94.4% and 94.45% accuracy

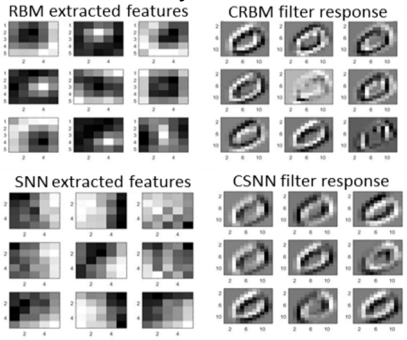


Fig. 8. Nine 5x5 extracted features and corresponding filter responses from our SNN and CRBM

TABLE I. Classification results

5x5 kernel (Network Size: 50x9x9)			7x7 kernel (Network Size: 98x9x9)		
Learning Time (NEC)			Learning Time (NEC)		
100	300	500	100	300	500
93.25	94.05	94.4	91.2	92.6	92.5
Fixed Point Precision (bits)			Fixed Point Precision (bits)		
8(4,4)	16 (8,8)	32 (16,16)	8(4,4)	16 (8,8)	32 (16,16)
90.3	91.35	93.25	89.85	87.45	91.2
Connectivity %			Connectivity %		
50	70	100	50	70	100
91.7	92.35	93.25	91.2	90.25	91.2

respectively using the 5x5 kernel, and 92.6% and 91.4% accuracy respectively with the 7x7 kernel. Please note that, although the state-of-the-art technique can recognize MNIST data with 99.2% accuracy [27], this is achieved using a multi-layer deep belief network with 60,000 training images. While ours has only one layer and trained using 2000 images. It is our next step to develop a multi-layer network using the stochastic SNN. The accuracy of our SNN is close to the best results in [21], which is 95%. However, [21] uses 6,400 excitatory neurons, which is 5 times more than ours. Furthermore, it memorizes the whole image, therefore it is hard to improve its accuracy by adding further layers; while ours is a convolutional network, which can be extended to a deep neural network.

Functional simulation of the hardware design was carried out to explore the tradeoff between cost and performance. TABLE I compares the accuracy of pattern classification when different fixed point data precisions and different connection ratio between the input and Bayesian layers are used. As we can see, the quality of learned features (i.e. the classification accuracy) drops marginally when the data precision is reduced from 32 bit to 8 bit. We also observed that losing 50% of the connection will not cause notable performance degradation for the 7x7 kernel. However, accuracy loss starts at 50% connections for the 5x5 kernel. Finally, we expedite training by reducing the time that the training image is exposed to the system, with only marginal impacts.

B. *Inference based sentence construction*

An inference network for sentence construction is created using the Bayesian neurons and stochastic SNN. It consists of lexicons representing words and phrases. As shown in Fig. 3, a lexicon is a subnetwork of Bayesian neurons for excitatory and ReLu neurons for inhibitory functions. Each Bayesian neuron represents a symbol, which in this case is a potential word or phrase at a certain location of sentence. The synapses between neurons across lexicons are created based on the log conditional probability of the two connected words (phrases). All neurons are initialized with the same intrinsic potential (i.e. the same initial firing rate). The most strongly connected neurons resonate and enhance each other and at the same time inhibit the other neurons in the same lexicons they belong to. The entire network settles on contextually correct associations and neurons with the highest firing rate in each lexicon marking the sentence that is grammatically correct and semantically meaningful. In this application, the inhibition layer performs the soft WTA function. It has an advantage over the hard WTA because symbols with lower excitation are not discarded, thus more information is retained during the inference. Fig. 9 shows the network topology.

We randomly picked 45 sentences from document images. Fuzzy character recognition is performed on these images which results in

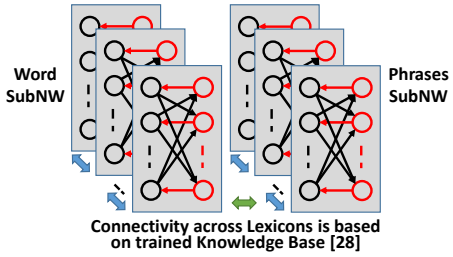


Fig. 9. Sentence confabulation network

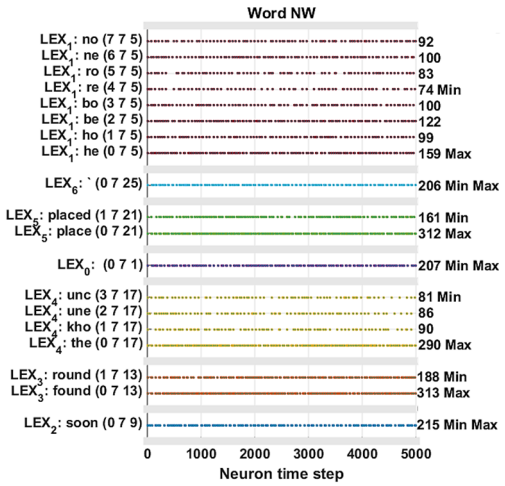


Fig. 10. Confabulation results raster plot

multiple possible words for each word position as described in [28]. For example, given input candidates [{he, ho, be, bo, re, ro, ne, no} {soon} {found, round} {the, kho, une, unc} {place, placed}], the SNN settles at a grammatically correct sentence as [he soon found the place]. Fig. 10 shows the raster plot for one of the sentences. The spike count for winning symbols is identified as "Max" in a lexicon. The stochastic SNNs are able to construct correct sentences for 83.8% of the test cases.

C. Hardware implementation analysis

An RTL design of the neuron model was developed and verified through functional simulation. The design is synthesized using 45nm, 1.1V CMOS technology. Our primary focus is to minimize power and area. The RTL design is synthesized with these constraints and we use the CACTI tool to estimate the access time and power consumption of the memory blocks with high threshold devices (for low leakage power.) In order to achieve similar speed as a biological neural system, our target is to complete one NEC in 0.5ms. This converts to a maximum clock period of 7.3ns. Synthesis results show that the minimum clock period is 3.15ns, hence we are well within the target margin with an area of 4460 μm^2 for digital logic and 209KB memory for 16 bit data path.

Not all resources are used every clock cycle. In our experiments typically the learning circuits are used 17% of time with sparse spiking pattern. Out of this, the multiplier is used for 3% of the time and the rest is spent on the adder for weight update. For recall operation the multiplier is utilized for 1.5% of the time. Overall the multiplier is utilized for only 2% of the time, therefore consuming little dynamic power.

VI. Conclusion

In this paper we have proposed a general-purpose, efficient and scalable Bayesian neuron model along with a digital logic design for pipelined implementation which is capable of in-hardware learning. The proposed model is simulated and validated using two different SNNs applications and compared with existing implementations.

References

[1] Ananthanarayanan, R.; Esser, S.K.; Simon, H.D.; Modha, D.S., "The cat is out of the bag: cortical simulations with 109 neurons, 1013 synapses," in *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on* , vol., no., pp.1-12, 14-20 Nov. 2009

[2] Gupta, A.; Long, L.N., "Character Recognition using Spiking Neural Networks," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, vol., no., pp.53-58, 12-17 Aug. 2007

[3] Behi, T.; Arous, N.; Ellouze, N., "Self-organization map of spiking neurons evaluation in phoneme classification," in *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on*, vol., no., pp.701-705, 21-24 March 2012

[4] Wang, Y.; Tang, T.; Xia, L.; Li, B.; Gu, P.; Yang, H.; Li, H.; Xie, Y.; "Energy Efficient RRAM Spiking Neural Network for Real Time Classification." In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pp. 189-194. ACM, 2015.

[5] O'Connor, P.; Neil, D.; Liu, S.C.; Delbruck, T.; Pfeiffer, M.; "Real-time classification and sensor fusion with a spiking deep belief network."*Frontiers in neuroscience* 7 (2013).

[6] Beyeler, M.; Dutt, N.D.; Krichmar, J.L.; "Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule." *Neural Networks* 48 (2013): 109-124.

[7] Maass, W., "Networks of spiking neurons: the third generation of neural network models." *Neural networks* 10, no. 9 (1997): 1659-1671.

[8] Sjöström, J.; Gerstner, W., "Spike-timing dependent plasticity." *Spike-timing dependent plasticity* (2010): 35.

[9] M. W. Oram, M. C. Wiener, R. Lestienne, and B. J. Richmond, "Stochastic nature of Precisely Time Spike Patterns in Visual System Neuronal Responses," *Journal of Neurophysiology*, vol. 81, pp. 3021—3033, 1999.

[10] H. S. Seung, "Learning in spiking Neural networks by Reinforcement of Stochastic synaptic Transmission," *Neuron,* Vol. 40, pp. 1063-1073, Dec., 2003.

[11] T. A. Engel, W. Chaisangmongkon, D. J. Freedman, and X. J. Wang, "Choice-correlated Activity Fluctua-tions Underlie Learning of Neuronal Category Representation," *Nature Communications* 6, Mar., 2015.

[12] Benuskova, L.; Kanich, M.; Krakovska, A.; "Piriform cortex model of EEG has random underlying dynamics." In *Proc. World Congress on Neuroinformatics. F. Rattay (Ed), ARGESIM/ASIM-Verlag, Vienna*, pp. 287-292. 2001.

[13] Arguello, E.; Silva, R.; Castillo, C.; Huerta, M., "The neuroid: A novel and simplified neuron-model," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, vol., no., pp.1234-1237, Aug. 28 2012-Sept. 1 2012

[14] Hines, M.; Carnevale, N., "The NEURON Simulation Environment," in *Neural Computation*, vol.9, no.6, pp.1179-1209, Aug. 15 1997

[15] Bower, J.M.; Beeman, D. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*. Springer Science & Business Media, 2012.

[16] Nessler, B.; Pfeiffer, M.; Buesing, L.; Maass, W., "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity." (2013): e1003037.

[17] Bichler, O.; Querlioz, D.; Thorpe, S.J.; Bourgoin, J.P.; Gamrat, C.; "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity." *Neural Networks* 32 (2012): 339-348.

[18] Furber, S.B.; Lester, D.R.; Plana, L.A.; Garside, J.D.; Painkras, E.; Temple, S.; Brown, A.D., "Overview of the SpiNNaker System Architecture," in *Computers, IEEE Transactions on*, vol.62, no.12, pp.2454-2467, Dec. 2013

[19] S. Park *et al*., "Neuromorphic speech systems using advanced ReRAM-based synapse," *Electron Devices Meeting (IEDM), 2013 IEEE International*, Washington, DC, 2013, pp. 25.6.1-25.6.4.

[20] J. M. Cruz-Albrecht, M. W. Yung and N. Srinivasa, "Energy-Efficient Neuron, Synapse and STDP Integrated Circuits," in *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 3, pp. 246-256, June 2012.

[21] Diehl, Peter U., Matthew Cook, Masami Tatsuno, and Sen Song. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity." *Frontiers in Computational Neuroscience* (2015).

[22] Benjamin, B.V.; Peiran Gao; McQuinn, E.; Choudhary, S.; Chandrasekaran, A.R.; Bussat, J.-M.; Alvarez-Icaza, R.; Arthur, J.V.; Merolla, P.A.; Boahen, K., "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations," in *Proceedings of the IEEE*, vol.102, no.5, pp.699-716, May 2014

[23] Schemmel, J.; Brüderle, D.; Grübl, A.; Hock, M.; Meier, K.; Millner, S., "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, vol., no., pp.1947-1950, May 30 2010-June 2 2010

[24] Merolla, P.A.; Arthur, J.V.; Alvarez-Icaza, R.; Cassidy, A.S.; Sawada, J.; Akopyan, F; Jackson, B.L.; et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface."*Science* 345, no. 6197 (2014): 668-673.

[25] http://white.stanford.edu/pdcwiki/index.php/Spike_generation_using_a_Poisson_process

[26] Taylor, G.; Cox, G.; "Digital randomness." *Spectrum, IEEE* 48, no. 9 (2011): 32-58

[27] Lee, H.; Grosse,R.; Ranganath, R.; Ng, A.Y.; "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 609-616. ACM, 2009.

[28] Qinru Qiu; Qing Wu; Bishop, M.; Pino, R.E.; Linderman, R.W., "A Parallel Neuromorphic Text Recognition System and Its Implementation on a Heterogeneous High-Performance Computing Cluster," in *Computers, IEEE Transactions on*, vol.62, no.5, pp.886-899, May 2013