Synaptic Sampling in Hardware Spiking Neural Networks

Sadique Sheik[1,2], Somnath Paul[1], Charles Augustine[1], Chinnikrishna Kothapalli[1], Muhammad M Khellah[1], Gert Cauwenberghs[2], Emre Neftci[3]
[1]Intel Corporation, Hillsboro, OR, USA
[2]Department of Bioengineering, UC San Diego, La Jolla, CA , USA
[3]Department of Cognitive Sciences, UC Irvine, Irvine, CA, USA Email: ssheik@mail.ucsd.edu

***Abstract*—Using a neural sampling approach, networks of stochastic spiking neurons, interconnected with plastic synapses, have been used to construct computational machines such as Restricted Boltzmann Machines (RBMs). Previous work towards building such networks achieved lower performances than traditional RBMs. More recently, Synaptic Sampling Machines (SSMs) were shown to outperform equivalent RBMs. In Synaptic Sampling Machines (SSMs), the stochasticity for the sampling is generated at the synapse. Stochastic synapses play the dual role of a regularizer during learning and an efficient mechanism for implementing stochasticity in neural networks over a wide dynamic range. In this paper we show that SSMs with stochastic synapses implemented in FPGA-based spiking neural networks can obtain a high accuracy in classifying MNIST handwritten digit database. We compare classification accuracy for different bit precision for stochastic and non-stochastic synapses and further argue that stochastic synapses have the same effect as synapses with higher bit precision but require significantly lower computational resources.**

I. Introduction

Biological synapses release neurotransmitters on the arrival of a pre-synaptic spike in order to induce a post-synaptic membrane potential. The effect of this transmitter release is represented as the synaptic weight in simulations of Spiking Neural Networks (SNNs). While it has been shown in biological recordings that this transmitter release is stochastic [1], most software simulations or hardware emulations of synapses do not exploit this stochasticity. Some of the primary reasons for not using stochastic synapses are: 1) lack of understanding of its implications 2) added cost for stochastic computation *e.g.* the generation of random numbers.

In this paper, we demonstrate that: 1) synaptic stochasticity can enhance the computational sampling capability of a SNN and 2) this added complexity can reduce the computational cost (compared to deterministic SNNs). We demonstrate these advantages within the context of a newly developed SNNs called Synaptic Sampling Machines (SSMs) [2]. SSMs are recurrent spiking neural networks inspired by Boltzmann Machines that exploit stochasticity in the synapses to perform synaptic sampling without the need of any other source of noise. We base our arguments on results obtained through an FPGA implementation of SSMs and a software simulation of this implementation. We use the SSM to perform a classification task on a standard MNIST hand written digits database [3].

In the following sections we describe how stochastic synapses can be implemented in digital systems. We describe the hardware implementation details for SSM and show that networks with stochastic synapses outperform networks with deterministic synapses at a classification task. Finally, we argue that networks with stochastic synapses can function with low bit-precision of synaptic weights, when compared to networks with deterministic synapses that achieved the same classification accuracy.

II. Materials and Methods

A. Synaptic Sampling Machines

SSM is a class of stochastic spiking neural networks inspired from Boltzmann Machines that is able to learn generative models of arbitrary data while potentially offering substantial improvements in terms of performance, power and complexity over existing methods for unsupervised learning in spiking neural networks [2]. As in neural sampling [4], spikes in the SSM are interpreted as Monte Carlo samples of a generative model that can be used for learning and inference. In contrast to neural sampling and previous studies, stochasticity in the network is induced by synaptic unreliability. The neurons are thus deterministic read-out units of its synaptic input samples. In this work, we consider synaptic blank-out noise: an action potential arriving at the synapse generates a post-synaptic response with a probability (p) smaller than 1. Synaptic unreliability plays the dual role of regularizer during learning akin to DropConnect [5] and an efficient mechanism implementing sampling in hardware, as we show below. Learning in the SSM is carried out by the STDP-based, event-driven Contrastive Divergence (eCD) rule [6] with nearest-neighbor spike-interactions. The training of this generative model is carried out in an unsupervised manner, involving multiple epochs of data presentation without any explicit reward or error signal. Using a network with a single hidden layer, SSM trained with eCD resulted in error rates of 4.4% on the MNIST hand-written digits task [2]. To date, the SSM is the best performing spike-based unsupervised learner on the MNIST task, while using a fraction of the neuron and synapse resources compared to previous models.

B. NSAT core architecture

Dedicated hardware emulation of SNNs can be extremely fast, power efficient and compact in comparison to general

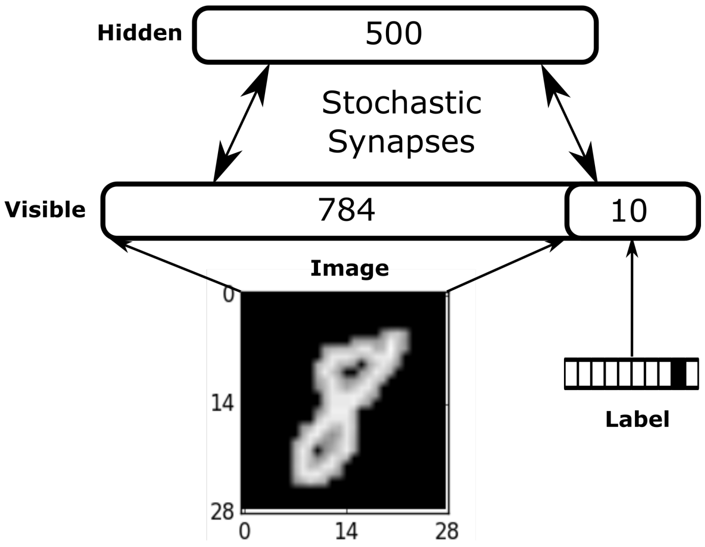

Fig. 1. Network architecture of SSM applied to classify MNIST image classification.

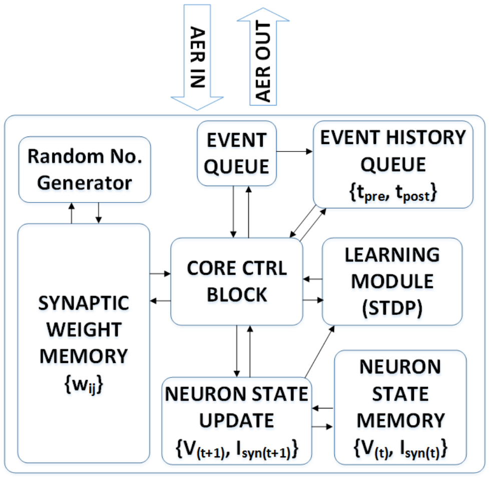
AER IN
AER OUT
Random No. Generator
EVENT QUEUE
EVENT HISTORY QUEUE {t_pre, t_post}
CORE CTRL BLOCK
LEARNING MODULE (STDP)
SYNAPTIC WEIGHT MEMORY {w_ij}
NEURON STATE UPDATE {V_(t+1), I_syn(t+1)}
NEURON STATE MEMORY {V_(t), I_syn(t)}

Fig. 2. Block diagram of Neuro-synaptic Array Transceiver (NSAT) architecture.

purpose processors. We propose Neural and Synaptic Array Transceiver (NSAT) architecture for a real-time implementation of SSMs and use an FPGA to prototype it. In hardware, stochastic synapses are implemented as part of a NSAT core. As illustrated in Fig. 2, the key components of the core are: i) *AER I/O:* Address Event Representation (AER) is the standard communication protocol followed at each core interface; ii) *Event Queue:* A FIFO for flow-control between the NSAT core and the fabric that interconnects multiple NSAT cores; iii) *Core Control Logic:* Mediates access to the weight memory and neuron states in the event of a spike; iv) *Synaptic Weight Memory:* A bi-directional crossbar [7] allowing both forward and backward look-up of neural synaptic weights; v) *Neuron State Update:* Carries out updates of the neural states in a time-multiplexed fashion. The logic stores the current neuron potential and synaptic current into a neuron state memory and retrieves it for updates during the next timestep; vi) *Event history queue:* Maintains a record of the firing times of the pre- and post-synaptic neurons. This block keeps a counter which is initialized with a specific time-window value when a neuron fires and decrements it at every timestep. The counter is essential for storing the eligibility of a neuron for synaptic weight potentiation and depreciation (*e.g.* for STDP); vii) *Learning module:* Implements the standard pairwise STDP rule. In the case of the eCD rule [6] used below, the learning rule used consists of a symmetric box window for $t_{pre} - t_{post}$ and nearest-neighbor spike-interactions.

C. Stochastic synapses

The synaptic weight memory unit in the NSAT core supports a low-overhead scheme for stochastic sampling of neural synapses. Synaptic sampling selectively samples/reads the synaptic weights based on random binary variables generated at the event of each pre-synaptic spike. Let us first consider deterministic synapses. During the training phase, the control block reads the weights in order to update the neuron potential and writes to them in case of a synaptic weight update. The STDP-based learning module captures the firing times of the pre/post-synaptic neurons from the event history queue and determines when a synaptic weight needs to be updated. For classification, the weights are read out at each pre-synaptic event if a connection exists. For the probabilistic scenario, i.e. stochastic synapses, the reading of synaptic weights depend on the value of a random binary variable generated for that synapse. A given synapse affects the post-synaptic neuron if this variable is 1. However if a synaptic weight update is required, as determined by the STDP module, then the weights are accessed for read and write, irrespective of the value for the random variable. In other words, synaptic weight update is always deterministic for the SSM model. During classification, learning is turned off, but pre-synaptic events are still stochastically gated by the random variable.

From the discussion above it is evident that connectivity information (1-bit) can be combined with the stochastic variable to achieve memory read power saving due to stochastic sampling. To realize this, we have implemented a novel crossbar hardware which incorporates both the connectivity bit and the weight bits at a single crossbar (Fig. 3 (a)). Fig 3 (b) shows the step where the connectivity bit (C-bit) is first read out by pre-charging the +ve and -ve bitlines (BL & BLB) for only the C-bit. A random binary pattern ($R_{pattern}$) is generated from exclusive-or combinations of the parallel outputs of two counter-propagating linear feedback shift registers [8]. The connectivity pattern ($C_{pattern}$) together with event history pattern ($E_{pattern}$) and a random binary pattern ($R_{pattern}$) is then used to read the weights from the crossbar array. Considerable savings in memory read power is achieved by preventing pre-charging of bitlines masked by the random binary pattern. Based on our measurements the savings outweigh the overhead in generating the random binary pattern. This stochastic sampling scheme was implemented in RTL and validated by mapping to a Virtex 7 Xilinx FPGA.

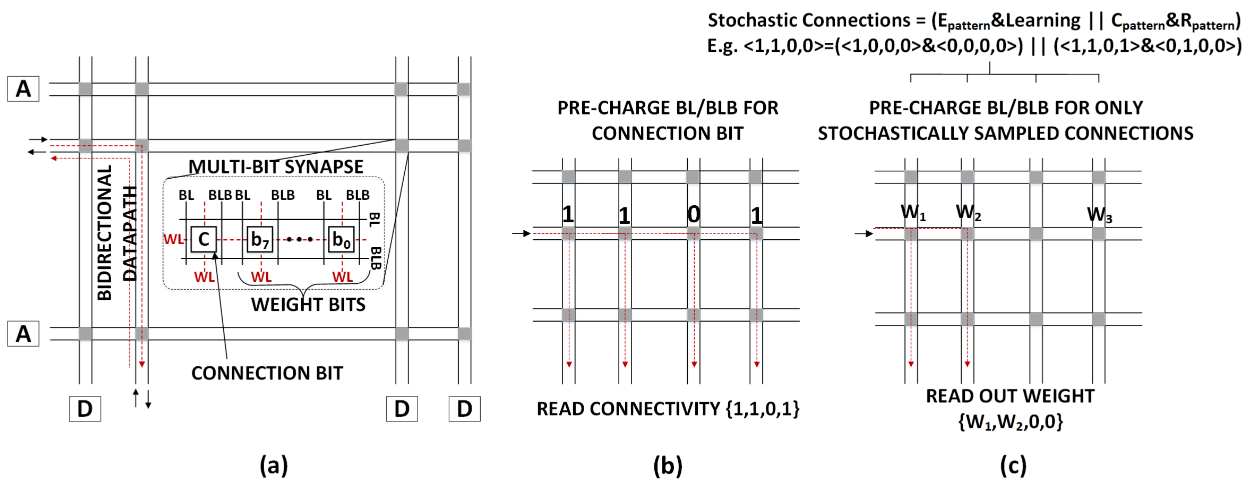
Stochastic Connections = (E$_{pattern}$&Learning || C$_{pattern}$&R$_{pattern}$)
E.g. <1,1,0,0>=(<1,0,0,0>&<0,0,0,0>) || (<1,1,0,1>&<0,1,0,0>)
A
A
MULTI-BIT SYNAPSE
BIDIRECTIONAL DATAPATH
BL BLB BL BLB BL BLB
BL BLB
WL
C
b$_7$
b$_0$
WL WL WL
WEIGHT BITS
CONNECTION BIT
D
D
D
PRE-CHARGE BL/BLB FOR CONNECTION BIT
1 1 0 1
READ CONNECTIVITY {1,1,0,1}
PRE-CHARGE BL/BLB FOR ONLY STOCHASTICALLY SAMPLED CONNECTIONS
W$_1$ W$_2$ W$_3$
READ OUT WEIGHT {W$_1$,W$_2$,0,0}
(a)
(b)
(c)

Fig. 3. a) Bi-directional crossbar for pre-to-post and post-to-pre synaptic weight look-up. Each crosspoint stores both 1-bit connectivity and an 8-bit weight; (b,c) Operations for synaptic sampling: b) First, for each pre-synaptic (input) spike, read the connectivity pattern $C_{pattern}$; c) then, mask connectivity with a random pattern $R_{pattern}$; and finally, access weights for crosspoints which remain enabled after masking.

D. SSM for MNIST hand written digit recognition

To characterize the effects of stochastic synapses, we trained SSMs to learn a model of the MNIST hand written digits (Fig. 1). The network consists of 794 Leaky Integrate and Fire (LIF) neurons in the visible layer (to accommodate 28×28 pixel images and 10 digit labels) and 500 neurons in the hidden layer. Each $100\,ms$ epoch during training consisted of 4 phases based on the eCD rule: $10\,ms$ of burn-in followed by $40\,ms$ of model learning phase, followed by $10\,ms$ of burn-in time followed by $40\,ms$ reconstruction phase. During classification, each epoch was $50\,ms$ and the label neuron with the highest firing rate was defined as the predicted label.

III. Results

A. Classification accuracy

In order to characterize the classification abilities of the SSM and validate its hardware implementation, we learned a model of the MNIST hand written digit image database for digit classification. We trained the network with an increasing number of training epochs and tested the classification accuracy against one thousand test images. Through the learning process we measure the classification accuracy for both stochastic and non-stochastic synapses. The results of this experiment are shown in Table I. The results show an increase in performance with increasing number of training samples with both stochastic and deterministic synapses. Consistent with [2], we observe an increase in classification accuracy with stochastic synapses. The highest classification accuracy was obtained at the end of training with 200K training samples. With deterministic synapses the accuracy was 78.9% whereas that for stochastic synapses (with $p = 0.5$), it was 90.8%, which is a 56% reduction in error rate with respect to deterministic synapses. Note that the original SSM reached 95.6% performance [2] (see Table I). This was because biases were also learned, the model was trained for longer (512k samples) and the learning rate decayed during the training.

TABLE I. MNIST classification accuracy of SSM. P is the probability of the stochastic synapses. While all the measurements were computed with no bias learning, the column with $p = 0.5*$ shows the accuracy with bias learning and demonstrates the optimal performance.

Training samples	p=1	p=0.75	p=0.5	p=0.5*	p=0.25
5k	0.515	0.684	0.72	0.812	0.603
10k	0.598	0.765	0.791	0.857	0.73
20k	0.71	0.802	0.817	0.878	0.799
50k	0.772	0.85	0.879	0.924	0.87
100k	0.793	0.875	0.896	0.941	0.883
200K	0.789	0.89	0.908	0.951	0.878

B. Power consumption

Stochastic sampling with synapses have a direct implication on power consumption with regards to memory access. While in a deterministic system every synaptic event requires a memory access in order to read its connection weight, a stochastically gated synapse with probability $p < 1$ only accesses the memory p times on average. We measured the total read and write accesses to the weight memory for the MNIST classification task described above. Table II lists the relative number of read and write operations on the weight memory for different synaptic probabilities and the corresponding relative power gain with respect to deterministic synapses. These statistics were collected for 50K training samples. During training, we observe an increase in total spiking activity of the network, with increasing values of p for stochastic synapses. Corresponding increase in memory access power was observed during the training process, the increment being marginal for lower probabilities ($p = 0.25$). The increase in memory power is primarily contributed by increase in memory writes during training phase. During classification phase, memory writes are absent and number of memory reads is lower. Memory power for stochastic synapses was therefore observed to be significantly lower compared to deterministic synapses ($p = 1$) and improve further with smaller probability values.

TABLE II. Effect of stochastic synapses on the number of memory read and writes events of the network.

	Training				Classification			Classification
	% Increase in				% Increase in			Error
p	Syn. Evs.	Rd	Wr	Mem. power	Syn. Evs.	Rd	Mem. power	
1.0	-	-	-	-	-	-	-	0.23
0.75	35.7	46	84.9	59	49	11.8	11	0.15
0.5	52.6	31.8	127.7	64	74.3	-12.8	-13	0.12
0.25	28.8	-22.8	74.8	10	70.2	-57.4	-58	0.13

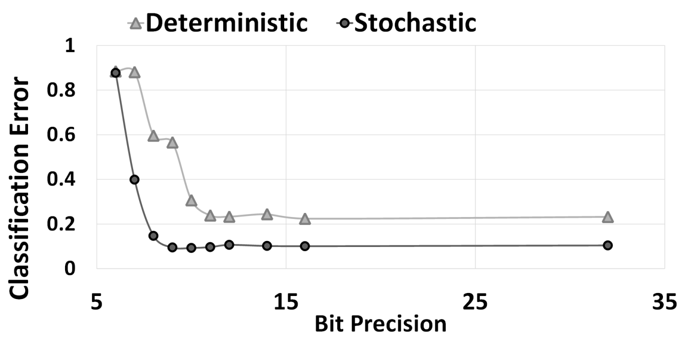

Fig. 4. Classification error for deterministic and stochastic ($p = 0.5$) synapses, trained with 50K samples. For the same number of bits, stochastic sampling provides higher accuracy. Alternatively, stochastic synapses with lower bit-precision achieves similar or better classification performance compared to deterministic synapses.

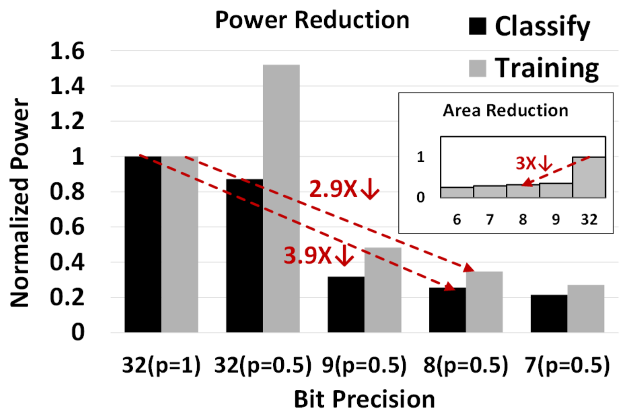

Fig. 5. SSM implementation allows scaling to lower bit-precision (8-bit) which in turn leads to significant area and power reduction compared to an iso-accuracy deterministic implementation with 32-bit precision.

C. Memory and Precision

Since stochastic synapses improve the classification accuracy, we explored the implications this would have on the bit precision of synaptic weights. From our experiments, we observe that for any given bit precision, synapses with stochasticity outperform deterministic synapses. Based on our measurements, for a classification accuracy of $\approx 80\%$ SSMs with stochastic synapses require 8 bit per synaptic weight compared to deterministic synapses that require atleast 32 bits (See Fig. 4). Our estimates at an advanced technology node show that this lower bit precision yields $3\times$ area savings for stochastic synapses as opposed to deterministic ones. There is also a corresponding decrease in the power consumption due to the smaller memory size. Based on the number of read and write operations during training and classification phases (Table II) and power estimates for a smaller and bigger memory, we see significant power savings of up to $3.9\times$ during classification and $2.9\times$ during training (ref to Fig. 5). Note that for iso-bit-precision, stochastic synapses ($p = 0.5$) do consume higher memory power compared to deterministic synapses owing to larger read/write activity (Table II). This power-saving at iso-accuracy suggests that stochastic synapses used for synaptic sampling virtually amplify the resolution of synaptic weights. Since a majority of silicon area in neuromorhpic SNN platforms is taken up by synaptic weight table, this is a very favorable result for design of compact neuromorphic hardware. Any reduction in the number of bits per synaptic connection weight has a direct impact on the silicon design overhead.

IV. Conclusion

In this paper we have presented the hardware implications of stochastic synapses in digital neuromorphic systems. Stochastic synapses were realized using random gating of valid connections stored in a bi-directional crossbar based synaptic weight memory. The proposed implementation yields considerable memory read power savings, particularly for lower values of probability. We also demonstrated that stochastic synapses require fewer bits for storage, which translates to significant area and power savings when compared to deterministic ones.

References

[1] P. Fatt and B. Katz, "Spontaneous subthreshold activity at motor nerve endings," *The Journal of physiology*, vol. 117, no. 1, pp. 109–128, 1952.

[2] E. Neftci, B. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs, "Unsupervised learning in synaptic sampling machines," *Arxiv*, Nov. 2015.

[3] Y. LeCun *et al.*, "The mnist database of handwritten digits," 1998.

[4] J. Fiser *et al.*, "Statistically optimal perception and learning: from behavior to neural representations," *Trends in cognitive sciences*, vol. 14, no. 3, pp. 119–130, 2010.

[5] L. Wan *et al.*, "Regularization of neural networks using dropconnect," in *Intl. Conference on Machine Learning (ICML)*, 2013.

[6] E. Neftci *et al.*, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in neuroscience*, vol. 7, 2013.

[7] J.-s. Seo *et al.*, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Custom Integrated Circuits Conference (CICC)*, IEEE, 2011.

[8] G. Cauwenberghs, "An analog vlsi recurrent neural network learning a continuous-time trajectory," *Neural Networks, IEEE Transactions on*, vol. 7, no. 2, pp. 346–361, 1996.