A Probabilistic Synapse with Strained MTJs for Spiking Neural Networks

Samuel Pagliarini, *Member, IEEE,* Sudipta Bhuin, *Member, IEEE*, Meric Isgenc, Ayan Kumar Biswas, Lawrence Pileggi, *Fellow, IEEE*

***Abstract*—Spiking neural networks (SNNs) are of interest for applications for which conventional computing suffers from the nearly insurmountable memory-processor bottleneck. This work presents a stochastic SNN architecture that is based on specialized logic-in-memory synaptic units to create a unique processing system that offers massively parallel processing power. Our proposed synaptic unit consists of strained magnetic tunnel junction (MTJ) devices and transistors. MTJs in our synapse are dual purpose, used as both random bit generators and as general purpose memory. Our neurons are modeled as integrate-and-fire components with thresholding and refraction. Our circuit is implemented using CMOS 28nm technology that is compatible with the MTJ technology. Our design shows that the required area for the proposed synapse is only 3.64 μm^2/unit. When idle, the synapse consumes 675pW. When firing, the energy required to propagate a spike is 8.87fJ. We then demonstrate a SNN that learns (without supervision) and classifies handwritten digits of the MNIST database. Simulation results show that our network presents high classification efficiency even in the presence of fabrication variability.**

***Index Terms*—Spiking Neural Network, Magnetic Tunnel Junction, Straintronics, Stochastic Synapse, Handwritten Digit Recognition**

I. Introduction

In the field of machine learning, the combination of growing data sets and (deep) learning schemes with neural networks has shown human level results in tasks such as recognition and classification [1], [2]. This performance level, however, comes at the cost of significantly increased demand for computational power.

The Von Neumann model [3] is employed in traditional computing architectures, which, in tandem with Moore's law, has provided an exponential reduction in the cost of computation. However, while significant increases have been seen in processor speed, significant overheads are incurred from growing memory hierarchies (caches) for leveraging the available processor speed: time and space locality dictates how efficient data can be accessed and at which energy cost. As CMOS scaling continues, however, it is apparent that for the current 14/16nm node and beyond that any further performance gain [4], [5] for logic will only increase the performance gap between the processor and the memory.

S. Bhuin and A. K. Biswas were with Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 15217, USA.

S. Pagliarini, M. Isgenc, and L. Pileggi are with Department of Electrical and Computer Engineering, Carnegie Mellon University.

Manuscript received May 11, 2018; revised Feb. 15, 2019.

Due to the aforementioned issues, a hardware solution that satisfies the processing requirements for machine learning must depart with the traditional computing paradigm. A promising solution is neuromorphic computation using hardware that is tailored for neural networks [6] and can present better energy efficiency.

Neural networks can be broadly classified into three different generations. The first generation is the perceptron, where the activation function is a Heaviside step function. In the second generation, many non-linear smooth activation functions such as sigmoidal, radial basis functions, and rectified linear unit became popular. These are widely used in back propagation learning of multilayer neural networks. Neurons of the third generation employ an activation function that integrates binary spikes over time, thus mimicking the sparsity of the previous generation in an efficient manner that lends itself well to efficient hardware implementation. These neurons are also referred to as integrate-and-fire neurons because of their summing and thresholding nature and have been used to build today's most complex neuromorphic hardware systems, such as IBM's TrueNorth [7].

The added temporal dimension allows spiking neurons [8], [9] to emulate the previous generation's neurons by encoding values as a spike rate, and thus they have been shown to be at least as computationally powerful as their first and second generation peers. The temporal dimension allows different encoding schemes [10], [11] and is very promising for tasks in which temporal information needs to be processed [12]. Currently, recurrent spiking neural networks (SNNs) are at the cutting edge of neuromorphic computing. However, very few have been physically implemented as circuits [7], [13]–[15]. Recently, novel beyond-CMOS designs have been explored to achieve better density and power efficiency [16]–[18], sometimes even at the cost of accuracy.

A cutting-edge neuromorphic circuit is difficult to demonstrate as a superior architecture when implemented purely in CMOS (i.e., with transistors). For example, the sheer number of synapses required for a few-node neural network requires intricate connections and routing, all of which can be relatively expensive to achieve with CMOS-only solutions. The use of a memristive crossbar [19] has been a topic of extensive research in the past decade. In such crossbar architecture, analog voltages are weighted and summed at the output node. Although the crossbar alone consumes low energy, the overhead due to repeated conversions from analog to digital (and vice-versa) can be expensive in terms of power and can require significant circuit area. Moreover, especially for a large scale crossbar, parasitic resistance reduces the accuracy of the network. Analog summation also suffers from mismatches and variations in the circuitry itself, such that the performance of a neural network with mismatched analog neurons is significantly impacted. On the other hand, fully digital implementations (that do not suffer from mismatch) cannot represent node values as a continuum of voltage values, therefore requiring many bits to represent the same data.

In this work, we describe a novel probabilistic synapse and demonstrate its use in a neural network that mitigates the aforementioned challenges. A magnetic tunnel junction (MTJ) that we have augmented to respond to mechanical strain from a piezoelectric material is utilized to create artificial synapses. These augmented MTJs are used to uniquely serve as both storage devices and random bit generators for the neural network. Of equal importance is the design of mostly-digital CMOS circuitry that is employed to overcome the mismatch and other non-idealities corresponding to fabricated MTJs.

The overall synapse circuit that is formed enables us to build a unique spiking neural network when combined with CMOS-only neurons. Importantly, the non-volatile storage provided by the MTJs allows for local access (i.e., there is no shared SRAM memory). When idle, there is effectively zero-power consumption from memory storage. Variation in the MTJ and CMOS portions of the synapses is assessed based on literature's measurement data for the former, and based on modeling of a commercial 28nm CMOS technology for the latter.

This paper is organized as follows: In Section II, the properties of augmented MTJs as true random number generators and memory are discussed. In Section III, the proposed neural network architecture is presented – implementation details of neurons and synapses are presented in subsections. Learning and process variation issues are also discussed in Section III. Our hardware implementation results and experiments are given in Section IV. Section V concludes the work.

II. MTJs: Background and Use

MTJs are magnetoresistive devices [20] that are typically created by employing an insulating oxide layer such as MgO sandwiched between two ferromagnetic layers, one free layer, and one hard layer. The spin polarization of the hard layer is pinned to a particular orientation by a pinning layer. The spin polarization of the free layer can be manipulated electrically, magnetically or by applying strain. For an elliptical in-plane magnetic device, the stable states are when the spin polarization of the free layer is at 0° or 180° angle with the major axis. If the spin orientation of free and hard layer are in the same direction, the effective resistance is R_P. The effective resistance is R_{AP} when the spin polarization of the hard layer and free layer are in the opposite direction.

This type of device is used to create non-volatile magnetic memories since these two states of the MTJ can effectively be used to represent ones and zeros. The energy barrier between these two states prevents spontaneous switching from one state to another. Spin-transfer-torque (STT) [21] switching mechanism is the most popular switching mechanism for MTJs. If the current density is above a critical current density, the magnetization of the free layer surmounts the energy barrier and a switch happens. Depending on the direction of the current, the magnetization switches from anti-parallel to parallel or from parallel to anti-parallel orientation, allowing the MTJ to assume the resistance values R_P and R_{AP}, respectively. The ratio of these two resistances is defined as Tunnel Magneto Resistance (TMR) and can be expressed as $TMR = (R_{AP} - R_P)/R_P$. These resistances as well as TMR itself are important characteristics of MTJs and have a direct impact on the neural network that we later demonstrate.

The augmented MTJ that we employ in our synapses is a Straintronic MTJ (S-MTJ) that operates with voltage generated strain, hence the energy requirement for writing is significantly lower than in other forms of MTJs. By rotating the magnetization of the free layer (either deterministically or randomly), the S-MTJ state can be altered. S-MTJs have been utilized as random number generators [22]–[24] as well as memories [25].

A. MTJs as True Random Number Generators

A reliable source of uncorrelated random noise is a fundamental component of any stochastic system. Our spike propagation is controlled by on-chip generated random bits. The quality of randomness and the degree of correlation across the chip affects the performance of the system [26]. A linear feedback shift register (LFSR) is the simplest form of a pseudo random number generator [27], [28] in terms of power and area consumption, although the periodicity is limited by the length of the LFSR. Recently proposed strained MTJs [24] are perfectly suitable for generation of low cost true random numbers that are uncorrelated.

S-MTJs are an extremely power efficient source of random bits since the state change is controlled by voltage generated strain. Unlike STT MTJs, switching is not deterministic for the S-MTJ. Fig. 1a shows an elliptically shaped nano MTJ that is delineated on top of a piezoelectric material. Here, W+ and W- are two write ports associated with the two electrodes on the piezoelectric material. R+ and R- are the read ports. In this particular configuration, W- is grounded and the write voltage is applied to W+ port. Due to inplane shape anisotropy, the MTJ is bistable along its major axis. When a voltage is applied to the W+ electrode, the MTJ settles at 90° orientation. When the voltage is removed from the W+ electrode, the MTJ goes into metastable state. Due to thermal noise, the MTJ then settles to either 0° or 180° orientation with a 50% probability. Fig. 1b shows the switching of an S-MTJ under voltage generated strain. The device is 70 nm $\times$ 55nm with a 1.625 nm thick oxide layer (MgO). Since the switching probability is 50%, the MTJ resistance will be R_{AP} or R_P with 50% probability. We highlight that the S-MTJ settles due to thermal noise and that no other input stimuli other than a short voltage pulse is required, granting the S-MTJ a desirable low-power writing characteristic.

An S-MTJ built with the properties described in [22] can operate at the same voltage level used for logic supply

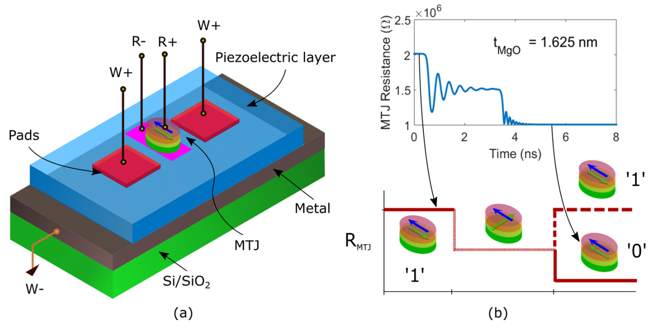

Fig. 1: (a) Schematic of a Straintronic MTJ. (b) Switching mechanism for TRNG under the application of voltage. Resistance has been calculated using a 1.625 nm thick MgO layer assuming a 100% TMR value [23].

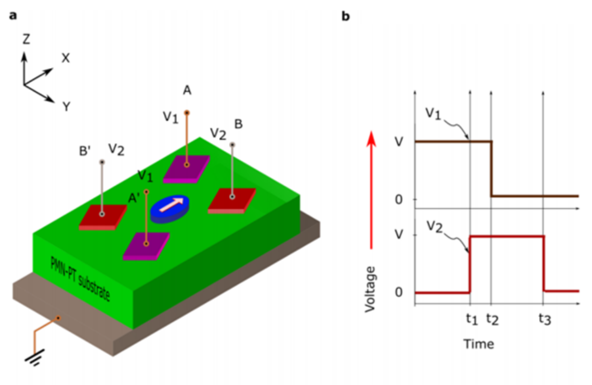

Fig. 2: Strain-induced complete magnetization reversal scheme. (a) An elliptical Co nanomagnet is fabricated at the intersection of the lines joining the centers of two pairs of electrodes delineated on a poled piezoelectric substrate whose bottom is grounded. (b) The timing diagram of the voltage pulses at the two electrode pair [30].

(VDD = 0.9V for the 28 nm technology used in this work), which allows for a seamless integration with CMOS. The operating frequency is limited by the switching time of the free layer of the MTJ under voltage generated stress. We calculate the worst case switching time to be $\approx$ 4.5 ns using the parameters mentioned in [22]. We also find the energy dissipation for writing to be approximately 1.4 fJ/bit. For STT-MTJs, the energy dissipation is typically in the order of 100 fJ/bit [29], further evidencing the low-power nature of S-MTJs.

The S-MTJs described in here would allow for a clock frequency of $\approx$ 222 MHz. The speed of individual devices is is not a limitation for the remainder of the circuit, as we will later show that our network has a global clock rate (slow) and a local clock rate (fast). The global clock rate is related to updating the synapses while the local clock rate is related to updating the neurons.

B. MTJs as Further Storage

S-MTJs can also be utilized as non-volatile storage units. With a modified configuration to act as a toggled memory, deterministic writing can be performed by toggling the stored bit. This type of toggle S-MTJs was proposed by [25] and experimentally demonstrated recently in [30]. Due to its voltage generated strain based writing methodology, storage S-MTJs still display low power consumption as when used as a random bit generator.

In this modified S-MTJ, two pairs of electrodes (A, A' and B, B') are delineated on top of a piezoelectric thin film, one at 30° and the other at $-30°$ angle with respect to the major axis of the MTJ as shown in Fig. 2a. The bottom of the thin film is grounded. Voltage pulses are applied on the two electrode pairs sequentially to modulate the energy profile and toggle the current magnetic orientation of the MTJ. This two-step method does not call for extreme precision, is practical and error-resilient, and works with any magnetostrictive material.

Fig. 2b shows the sequence of voltage pulses that has to be applied to toggle the magnetic orientation. When a positive voltage V1 (= V) is applied between the electrically shorted first pair AA' and ground, the MTJ is switched by the one-third of the distance towards the opposite direction. Next, a positive voltage V2 (= V1 = V) is applied between the second pair BB' and ground at time t1 and simultaneously turning off the voltage (V1) at pair AA' after a time t2. The MTJ now rotates by another one-third of the distance towards the opposite state. When all the voltages are turned off at t3, the MTJ switches completely to the opposite state. This performs the writing of one bit (say '1') from the other bit (say '0'). There is no critical precision demand on t1, t2 or t3, which makes the scheme practical to implement.

The only caveat of this S-MTJ configuration is that it has to be read first such that a known bit can be written (i.e., by deciding to keep the bit or to toggle it). This caveat would impact a learning scheme, where it is expected that synaptic weights will change dynamically. However, it has no impact when the network is performing inference.

III. Neural Network Architecture

A feedforward SNN is used as a platform to demonstrate the efficiency of the proposed synapse. The network architecture is shown in Fig 3. A two-layer architecture is used, with an encoding layer and a output layer that performs classification.

The encoding layer takes the input image and generates a Poisson spike train with a rate that is proportional to the pixel values. The encoding layer may well be a sensor interface converting an analog input to digital spikes. Excitatory feedforward synapses (shown as blue solid lines in Fig. 3) propagate the spikes from the encoding layer to the classification layer. Neurons of the classification layer are laterally connected to each other by inhibitory connections (shown as orange solid lines in Fig. 3). Thus, each neuron tries to inhibit the others once it fires a spike, a desirable feature for a network that performs classification.

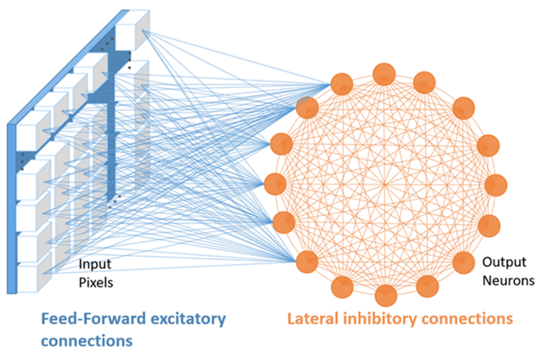
Input
Pixels
Output
Neurons
Feed-Forward excitatory connections
Lateral inhibitory connections

Fig. 3: Network configuration: A feedforward SNN. The encoding layer generates a Poisson spike train while the output neurons have lateral inhibitory connections.

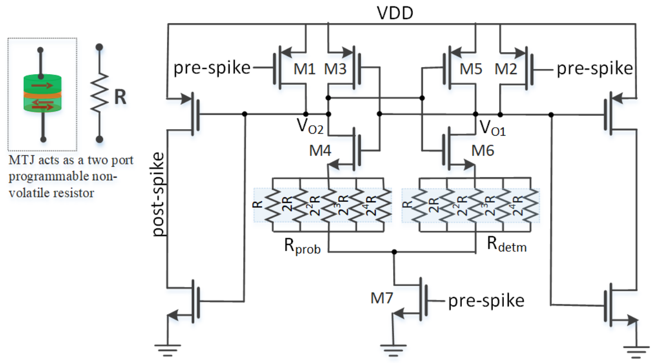

Fig. 4: Proposed synapse. The cross-coupled latch functions as a logic-in-memory element that compares R_{prob} to R_{detm}. The value of R_{detm} is determined by the MTJs that act as memory. The value of R_{prob} switches in a truly random manner.

For the sake of clarity, we classify spikes into pre-synaptic and post-synaptic. A generated spike, regardless of the layer it originates from, is called a pre-synaptic spike. After the spike event is processed through the proposed synaptic unit, it is called a post-synaptic spike. Post-synaptic spikes are then accumulated at the neurons. Note that the synapse is inactive unless a pre-synaptic spike occurs. A pre-synaptic spike activates the synapse and is propagated with a probability that is proportional to the weight of the corresponding synapse [31]. Synapses are programmed with parameters learned from an offline spike time-dependent plasticity (STDP) learning scheme. STDP learning is generally considered hardware-friendly and can be implemented on-chip for online learning. We next describe the hardware implementation of the synapse and neuron units utilized in our architecture.

A. Proposed Probabilistic Synapse Unit

Our proposed synapse is a stochastic synapse, meaning that the synaptic propagation is modulated stochastically by the weight of the synapse. The output or the post-synaptic spike event, S_{out}, can be denoted by

$$S_{out} = \omega \times S_{in} \tag{1}$$

where S_{in} is the incoming or the pre-synaptic spike event and ω is the weight value of the corresponding synapse. In hardware, the multiplication of the weight with the pre-synaptic spike can be emulated by an 'and' gate. Let us now discuss how the weight of an S-MTJ-based synapse is determined.

In a traditional hardware implementation of a neural network, weight coefficients would be stored as digital bits in memory (preferably on-chip); each neuron is then associated with a range of memory addresses that represent the synapses it is connected to. These addresses would be fetched on demand from the memory. We instead store the weights in S-MTJs that are local to the synapse and directly generate post-synaptic spikes using a simple cross-coupled latch [32].

In this work, S-MTJs are configured in a manner so that they can be either programmed to store weights or to generate random numbers, both of which are critical for our proposed SNN. We will refer to these as Deterministic MTJ and Probabilistic MTJ, respectively. The combined S-MTJ and latch circuit has been proposed as a low power and efficient source of random bits in [22], [24], which performs all the operations in one simple circuit in the analog domain. The proposed latch is shown in Fig. 4. The resulting output is digital and ready to be used by the rest of the system. The latch functions by comparing the equivalent resistances from the Probabilistic MTJs (R_{prob}) and from the Deterministic MTJs (R_{detm}). The MTJs are sized and binary-weighted to represent a desired range of resistances. R_{prob} and R_{detm} are parallel combinations of 5 binary-weighted Probabilistic MTJs and Deterministic MTJs, respectively.

The resistance of an MTJ is inversely proportional to the area of its elliptical cross section. By changing the length of the major and minor axis, the MTJ resistances can effectively be modulated to be binary-weighted. The equivalent resistances R_{prob} and R_{detm} are compared by the cross-coupled latch at the rising edge (from '0' to '1') of the pre-synaptic spike. As shown in Fig. 4, the equivalent conductances of the set of Probabilistic MTJs and of the set of Deterministic MTJs are

$$\begin{aligned}\frac{1}{R_{prob}} &= \frac{1}{P_0\times R}+\frac{1}{P_1\times 2R}+\frac{1}{P_2\times 2^2R}+\frac{1}{P_3\times 2^3R}+\frac{1}{P_4\times 2^4R}\\ \frac{1}{R_{detm}} &= \frac{1}{D_0\times R}+\frac{1}{D_1\times 2R}+\frac{1}{D_2\times 2^2R}+\frac{1}{D_3\times 2^3R}+\frac{1}{D_4\times 2^4R}\end{aligned} \tag{2}$$

$$\begin{aligned}P_n &= (TMR+1) & if\, n^{th}\, bit = 1\\ &= 1 & if\, n^{th}\, bit = 0\\ D_n &= (TMR+1) & if\, n^{th}\, bit = 1\\ &= 1 & if\, n^{th}\, bit = 0\end{aligned} \tag{3}$$

where R_{prob} and R_{detm} represent the total resistances of Probabilistic and Deterministic MTJs, respectively. R is the parallel spin polarized MTJ resistance (R_P), and TMR is the tunnel magneto resistance of the MTJs. Depending on the spin polarization of the free layer with respect to that of the hard layer, the effective resistance of the MTJ will be either R_{AP} or R_P. The n^{th} bit of the probabilistic array or the deterministic array is denoted by '1' when the MTJ resistance is R_{AP} and '0' when MTJ resistance is R_P. As shown in Fig. 4, at the rising edge of the pre-synaptic spike or 'pre-spike', the output V_{O1} will be either 0 or 1 based on the conductance at both sides.

The latch is activated when the pre-synaptic spike (denoted as 'pre-spike') signal is logic '1'. Otherwise, the latch is in standby mode consuming negligible power. This method is specifically suitable for SNN as the spikes are sparse in nature. When 'pre-spike' is logic '0', the output nodes (V_{O1} and V_{O2}) are pre-charged to VDD by the PMOS transistors M1 and M2. When 'pre-spike' goes from logic '0' to logic '1', the bias transistor (M7) switches on, and the pre-charged transistors (M1 and M2) turn off. Based on the resistance value on either side, the rate of falling of the voltages on the two output terminals (V_{O1} and V_{O2}) will be different. The side that has lower resistance will fall faster than the other side, and whenever one node falls below the switching threshold of the cross-coupled inverter, positive feedback latches it in that direction. For example, if $R_{prob} > R_{detm}$, at the rising edge of the 'pre-spike', V_{O1} discharges faster than V_{O2}, and when V_{O1} falls below the switching threshold of the cross-coupled inverters, M3 turns on and recharges V_{O2} to VDD and M6 discharges V_{O1} to ground.

In summary, the weight of the synapse is a function of the conductances of both sides. If the programmed weight of the Deterministic MTJs is greater than the random number of the Probabilistic MTJs, a pre-synaptic spike propagates through the synapse. Otherwise, a pre-synaptic spike is not propagated. If the pre-synaptic spike is zero, the latch is always pre-charged and the post-synaptic spike is zero, irrespective of the weights.

It should be noted that the write paths of the S-MTJs are electrically insulated from read paths, thus writing can be performed irrespective of the state of the latch.

B. Neuron Unit

We use a simplified Izhikevich neuron [33] in our SNN. The integrate-and-fire neuron shown in Fig. 5 sums all the incoming spikes and produces an outgoing spike when a threshold is surpassed. After a neuron successfully generates a spike, the membrane potential is restored to the resting potential and remains inactive for a specified refractory time. Our neuron implementation also accounts for an optional leak rate to be subtracted from the membrane potential periodically.

In Fig. 5, the synaptic array represents the post-synaptic spikes associated with each neuron. The post-synaptic spikes are processed with a local high speed clock sequentially, one after another. The spike counter consists of an up/down counter. Each spike activates the up/down counter while the event controller activates the up/down knob based on the address of the incoming spikes. The feedforward spikes are added to the counter while the lateral inhibitory spikes are subtracted from the counter. The counter output acts as the membrane potential (v_{mem}) of the neuron. The membrane potential is compared with a threshold (v_{thr}), and a spike is generated based on the comparison. Upon firing, the neuron

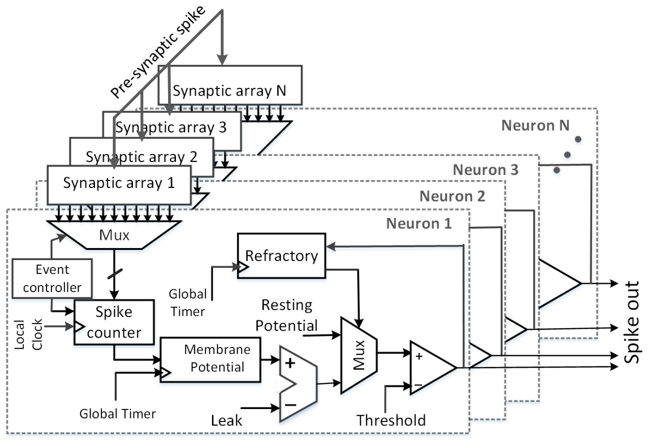

Fig. 5: Block diagram of a simplified integrate-and-fire neuron. The neuron implementation is fully digital.

enters into a refractory state, and the membrane potential restores the resting potential (v_{rest}). The simplified spike response model for the neuron is expressed below.

$$\begin{aligned} & if \quad v_{i,mem}(t) > v_{thr} \\ & S_i(t) = 1; \quad if \quad t > (t_{n-1} + t_{refrac}), \\ & v_{i,mem}(t+1) = v_{rest} \end{aligned} \tag{4}$$

where $S_i(t)$ is the spike response at time t, t_{n-1} is the timing of the last spike occurrence at the i^{th} neuron, $v_{i,mem}$ is the membrane potential of i^{th} neuron at time t, v_{thr} is the threshold voltage, and t_{refrac} is the refractory time.

The membrane potential of output neuron 'i' at time t is

$$v_{i,mem} = \sum_{j=1}^{n} \omega_{j,i}^{ff} S_j - \sum_{k=1}^{m} \omega_{k,i}^{lat} S_k - v_{leak} \tag{5}$$

where $\omega_{j,i}^{ff}$ is the excitatory feedforward connection weight parameter from input node 'j' to output node 'i', $\omega_{k,i}^{lat}$ is the inhibitory lateral connection from node 'k' to node 'i', S_j and S_k are pre-synaptic spike events, n is the number of input neurons and m is the number of output neurons.

The timing diagram of the neuron operation is shown in Fig. 6. Each neuron processes the post-synaptic spikes sequentially, followed by application of leak and threshold. A spike is generated based on the membrane potential and a pre-synaptic spike for the next cycle is sent to the synapses of the output neurons as described in the previous subsection. The proposed synaptic latch has to be pre-charged first by applying a reset to the 'pre-spike' signal at each global cycle.

C. Spike Time Dependent Plasticity

STDP has been a popular method of learning in SNNs due to its locally adaptive nature and biological plausibility. We applied inhibitory and excitatory STDP [34]–[36] to train a large network that utilizes our proposed synapses. These results are later reported in Section IV-D.

One of the challenges of STDP is to select a learning strategy and learning rate that leads to high quality of classification. When considering online learning, specialized

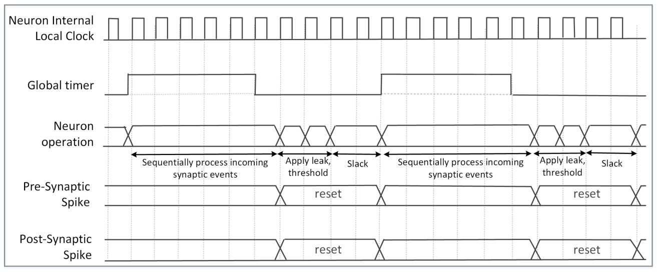

Fig. 6: Each neuron processes the incoming synaptic events sequentially at a high frequency (local clock). The neuron keeps processing and accumulating for as many cycles as incoming synapses it is connected to. Thresholding is then applied. A reset is applied at a slow frequency (global clock) to allow the latches to be pre-charged. The neuron operation is trivially implemented with a finite state machine.

structures have to be devised to keep track of the STDP events (i.e., correlated pre-synaptic and post-synaptic spikes) and the evolution of weights. These structures are rarely useful for inference, and therefore impose a heavy overhead.

In our network, the input nodes are connected to the neurons through excitatory synapses, and the neurons are inhibited by each other through inhibitory synapses (see Fig. 3). Counters can be utilized to store STDP events [13]. At each clock cycle, these learning counters perform a check for pre- and post-synaptic spikes over a pre-defined window of time. This operation has to be repeated over many clock cycles due to the sparse nature of spikes.

After a training epoch is complete, the Deterministic MTJs can be rewritten with new values before the next training data is analyzed. Once the SNN has learned enough from input data, the weights converge and each neuron of the output layer becomes selective of only one class. Therefore, we achieve unsupervised classification as the input data is not labelled.

Hardware realization benefits from synaptic weights that can be represented with a small amount of bits, as it leads to small area and power. However, there is a compromise between the number of bits and the learning of complex patterns. In Fig. 4, we have purposefully drawn 5 resistors in parallel as we determined that the associated range of resistances can be feasibly fabricated today. In Section IV-D, we show that learning is affected by this decision.

Finally, we highlight that all the weights of our proposed SNN are positive, therefore it is not necessary to use an extra bit for storing the weight sign.

D. MTJ and CMOS Process Variation

Significant process variation occurs for scaled CMOS technologies whereby transistor dimensions, oxide thickness, and threshold voltage are all possibly affected. These variations shift the operation of the circuit from its expected/nominal behavior. Considering the operation of the latch described in Section III, it is apparent that mismatch-induced variation is concerning due to the differential nature of the circuit. MTJs suffer from process variations as well, mostly due to variations in size and oxide thickness [37].

The effect of the combined CMOS and MTJ process variations has to be taken into consideration for an individual synaptic unit as well as for a complex network with many synapses. The imperfections of CMOS transistors and MTJs impact learning and inference, albeit in different ways and degrees. In the next Section, we detail a series of experiments we performed in order to assess and model the undesired effects of process variation.

IV. Experiments and Results

A. Synapse Implementation

As previously mentioned, the proposed synapse has 5 Deterministic MTJs and 5 Probabilistic MTJs. The MTJs are arranged in parallel fashion. We assume a baseline resistance of 10KOhm for each MTJ. Therefore, when a synapse is programmed with the value '10000', the equivalent resistance is $R_{eq} = 20K||20K||40K||80K||160K$ since the resistance values are binary-weighted.

Once the individual MTJ resistance is known, the range of possible equivalent resistances seen by the latch circuit is also known. The synapse CMOS circuitry can then be optimized by performing transistor sizing. Transistor sizing determines the node to node relationships that guarantee the dual-purpose of the circuit, i.e., working as a latch and as a sensor. Transistor sizing also determines the robustness of the latch with respect to process variation. In the CMOS technology utilized in this work, the minimum pre-shrink size of a transistor is L=30 nm and W=100 nm. Our initial implementation had all transistors sized as X1 (i.e., minimum allowed size).

However, when X1 transistors are employed in a differential pair like the one present in our latch, mismatch becomes a concern. Employing larger (wider) transistors is a well-known strategy to minimize the effects of mismatch. The transistor characteristics of our optimized synapse are given in Table I. In this configuration, the synapse area is only **3.64 μm^2**.

TABLE I: Transistor sizing of the synapse.

Transistor ID	**Type**	**Sizing (width)**
M1, M2	PMOS	3X
M3, M5	PMOS	3X
M4, M6	NMOS	2X
M7	NMOS	2X

In Table II, we provide power and energy characteristics of the synapse when no MTJs are being written. The CMOS portion of the synapse is sized according to Table I. We note that leakage power is the power consumed by the synapse when in stand-by mode since no pre-synaptic spike is occurring. Leakage power varies with temperature as highlighted in In Table II. Conversely, the energy per spike is measured assuming a pre-synaptic spike is taking place. These values correspond to a synapse that is not equipped with learning circuitry, i.e., it can only perform inference.

TABLE II: Power and energy characteristics of the synapse.

Corner	Leakage power (pW)	Energy per spike (fJ)
Slow-Slow, 125C	2850.1	1.70
Typical, 25C	675.6	1.87
Fast-Fast, -40C	514.9	1.83

In addition to the energy dissipated per spike, the synapse dissipates power when writing to the MTJs. Deterministic MTJs are written only once if the network has been trained offline. Due to the non-volatility of the MTJs, the weights are persistent and do not need a 'refresh'. Therefore, we consider this energy to be negligible. On the other hand, Probabilistic MTJs have to be written periodically during runtime. This periodicity is determined by the global clock rate that is much slower than the local clock rate. The energy required to randomly switch all Probabilistic MTJs in one synapse is approximately 7fJ. Accounting for the subsequent computation performed by the latch itself, a synapse may dissipate as much as 8.87fJ per propagated spike.

B. Neuron Implementation

In the analog domain, the integration or accumulation of post-synaptic spikes at the neuron is often done by a charge pump that stores charge in a capacitor. A leaky behavior can be present due to the inherent parasitics or any purposefully created current path. On the other hand, in a digital neuron, integration can be achieved with a counter that sums the incoming spikes. A digital comparator can be used to compare the sum with a threshold and fire accordingly. Additional circuitry is also required for modeling the refractory period and the leak.

Moreover, in any advanced CMOS technology, an all-digital implementation of neuron is beneficial since it does not suffer from mismatch, while presenting an ease of implementation with automated synthesis tools. Hence, a fully digital approach has been taken to implement our neuron using a commercial 28nm CMOS technology.

The neuron is designed using industry standard flows. The hardware description is implemented in Verilog using a finite state machine that captures the neuron cycle-by-cycle operation (i.e., to integrate, apply leak, apply threshold, consider refraction, and to finally start over). Logical synthesis, placement, clock tree synthesis, and routing were performed using foundry-provided standard cells. The power and area estimations are a combination of the reports from the synthesis tools and the estimation for the devices described in this paper. We also assumed a backend integration process where the MTJs could fit on top of the CMOS circuitry. The circuit has a global clock frequency of 83 MHz, i.e., the spike generation frequency is 83 MHz for each neuron, while the internal clock runs at 830 MHz. Each all-digital neuron dissipates approximately 1mW while operating at 830 MHz and takes an area of approximately 1400 μm^2. The gate count for each neuron is approximately 1K gates.

C. Process Variation

Since the neurons of our network are digital, it is only the synapse that has to be carefully designed with respect to process variation. We analyzed the behavior of an individual synapse under variation modeled from two sources:

- Process variation on the CMOS circuitry
- Resistance variation on the MTJ device

For the first source of variation, transistor mismatch creates an offset at the source nodes of the cross-coupled inverters of the synaptic latch. The input referred offset of the latch biases one of the resistive arrays. We have performed extensive Monte Carlo simulations following the guidelines recommended by the foundry. In summary, it means that the CMOS circuitry of the synapse was SPICE simulated at different (global) corners for fast, typical, and slow conditions, but also at varying degrees of mismatch (local). A total of 10000 Monte Carlo runs were performed, which is more than sufficient for a circuit with a handful of transistors like the proposed latch. We do not introduce any MTJ resistance variation at this time in order to decouple the results. We then programmed the synapse with resistance values of R_{prob} = 10KOhm and R_{detm} = 5KOhm. It is expected that, in ideal conditions, the synapse would spike 100% of the time whenever presented with a pre-synaptic spike. However, we verify that CMOS imperfections can sometimes overcome a large difference in MTJ resistance, as detailed in Table III.

Mismatch-induced offset can be minimized by increasing transistor width and/or length. Instead of using minimum sized transistors, we resort to using oversized transistors as detailed in Table I. We present results for the minimum sized synapse as well as for the optimized version which employs 2X and 3X transistors. It is evident that increasing the transistor size leads to a more robust implementation, with an error rate in the order of only 2.3%. Further increasing the transistor sizes leads to minimal improvement in robustness, while incurring a significant penalty in area.

TABLE III: Effect of CMOS variation on the proposed synapse.

Transistor sizing	# spikes	# no spikes	Error rate
min (1X)	9267	733	7.3%
optimized (2/3X)	9772	228	2.3%

As to variation on the MTJ devices, we highlight that it presents itself as TMR variation (i.e., ratio of R_P to R_{AP}) and resistance variation (i.e., ratio of R_{prob} to R_{detm}). Any variation in TMR affects the system significantly when the TMR value is high[1] ($\gg 1$). We, therefore, simulate the synapse with a low TMR (=1) in order to analyze the effect of the resistance variation.

[1]For a high TMR value, even a 10% variation can be significant and nearly insurmountable to overcome. Although, with improved process fabrication, the variations are expected to be much lower. The variation of MTJs in industry production has been minimal in recent days. Cross-wafer and wafer-wafer variability have been reduced to <1%. The sigma of the distribution in MTJ resistance is around 1.5%. A detailed discussion of MTJ variation is provided in [37].

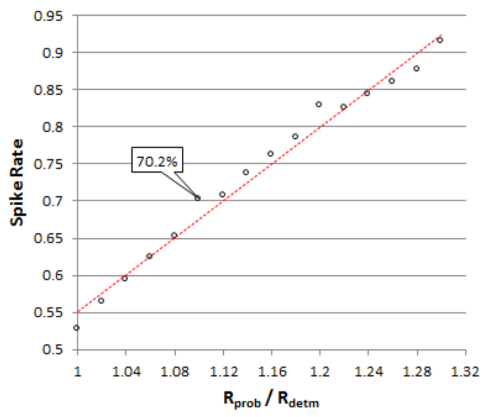

Fig. 7: Synapse spike rate versus ratio of R_{prob} to R_{detm} for TMR = 1, R_{detm} = 10KOhm and R_{prob} is varied.

In Fig. 7, we plot the spike rate of a synapse versus the resistance variation. We assume both resistances have a nominal value of $R_{detm} = R_{prob}$ = 10KOhm and apply variation to R_{prob}. When the ratio is 1, the synapse should have a probability of firing of 50%. Given the results shown in Table III, the non-idealities in the CMOS circuitry increase the spike rate to approximately 52.8%. When the ratio is greater than 1, the synapse should fire more often. A (positive) variation of 10% in R_{prob} increases the firing rate to approximately 70% as indicated by the callout on Fig. 7. Each point represents the spike rate averaged over 10000 Monte Carlo runs. Thus, a total of 160000 Monte Carlo simulations were utilized to plot Fig. 7.

We only plot the cases where R_{prob} is greater than R_{detm}, but the behavior is nearly identical for the opposite cases (i.e., R_{detm} greater than R_{prob}). The trendline present in Fig. 7 could be extended to represent the opposite cases as well.

In the following section, we present simulation experiments that are performed on both ideal and non-ideal neural networks. It must be noted that while Monte Carlo simulations can be performed at circuit-level, the approach **does not scale** to block- or chip-level simulations. It is then necessary to model the CMOS variation and MTJ variation as compact models that can be incorporated into a larger SNN simulation. In the analysis that follows, we assumed the variation in MTJ resistances to be a Gaussian distribution with a mean of 10KOhm and (pessimistic) standard deviation of 10%. For the CMOS variation, we devised a compact model that utilizes a piecewise linear equation that captures the probability of firing a spike when a spike is not expected to occur. The equation has 6 pieces (all values are in KOhms):

1) R_{prob} in [4, 6], $R_{detm} \approx 5$, $R_{prob} \leq$ Rdet
2) R_{prob} in [4, 6], $R_{detm} \approx 5$, $R_{prob} >$ Rdet
3) R_{prob} in [6, 9], $R_{detm} \approx 7.5$, $R_{prob} \leq$ Rdet
4) R_{prob} in [6, 9], $R_{detm} \approx 7.5$, $R_{prob} >$ Rdet
5) R_{prob} in [8, 12], $R_{detm} \approx 10$, $R_{prob} \leq$ Rdet
6) R_{prob} in [8, 12], $R_{detm} \approx 10$, $R_{prob} >$ Rdet

These cases are necessary since the effect of the CMOS variation on the spike rate is modulated by the range of the MTJ resistances and their difference. We use three ranges, centered around 5, 7.5, and 10KOhms.

D. Handwritten Digit Recognition

An SNN that performs digit recognition was built using the synapses and neurons previously described. The input to the SNN is an image with 784 pixels (28x28). Each grayscale pixel assumes a value between 0-255. The MNIST database [38] of handwritten digits has been used to train and test the SNN. The database consists of a total of 60000 training images and 10000 test images.

We make use of the BRIAN simulator [39] to describe our network dynamics. We use biologically plausible ranges for almost all of the network parameters. The authors of [40] performed a study on STDP strategies and their associated classification accuracy. We make use of their findings and parameters to train our network[2].

In order to assess the learning capability of our SNN, we first build a network with 100 excitatory neurons and perform offline training using ideal stochastic synapses. Notice how this configuration leads to a neuron being connected to 784 excitatory synapses and 99 inhibitory synapses. In order to compensate for this imbalance, the Poisson spike train that represents the input data is dynamically adjusted. During training, each input image is presented 3 times to the network. In Fig. 8, we plot the weights of the excitatory synapses after 2000 images are presented to the network (i.e., 6000 training samples total). Each digit in the image corresponds to 784 synapses organized as a 28x28 matrix. The image is further arranged as a 10x10 matrix, where synapses are grouped according to the neuron they are connected to. It is clear that the network has already learned several patterns, but the strength of the weights could still be improved.

However, if we were to perform online learning, the ability of the network to learn is hindered by the imperfections of the MTJs and of the CMOS circuitry. We introduce these two components as noise in the BRIAN simulation. The 'noise frequency' is determined by the modelling we described in Section IV-C. Our results are shown in Fig. 9, where we provided the network with 6000 and 12000 training samples. The noise introduced in the simulation delays the convergence of the learning procedure. The result in Fig. 9 (a) highlights how several digits are still noisy, as if no training had been performed at all. By continuing to provide more training samples, the weights start to converge as shown in Fig. 9 (b). Notice that even when twice the amount of training samples were provided, Fig. 9 (b) still presents many blurry digits like the one highlighted in Fig. 9 (c), whereas Fig. 8 presents very few of those.

We proceed by measuring the classification performance of the network during inference. We assume that training was performed offline and on an ideal network. By doing so, we avoid the loss of precision and increased training time that is highlighted in Fig. 9. While learning was performed offline and on ideal hardware, the double-precision weights have to be mapped to the 5-bit synapses, i.e., they become

[2]The authors of [40] share their code and training parameters on https://github.com/peter-u-diehl/stdp-mnist

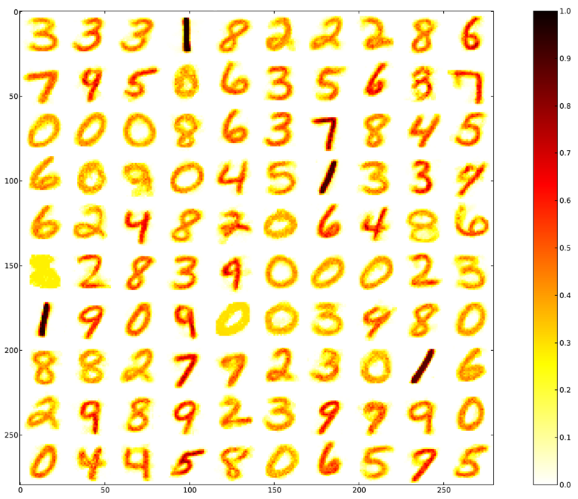

Fig. 8: Weights of the excitatory synapses after only 6000 training samples are provided. Most set of synapses already settled to a clear prototypical input. The scale on the right hand side represents the weight strength. The darker a pixel is, the stronger the synaptic weight.

quantized between 0 and 31. This quantization leads to a loss of accuracy as shown in Fig. 10. We trained the networks composed of 100 and 400 neurons using all the 60000 images of the MNIST database, each image being presented 3 times. During inference, we used only the 10000 images from the test set. We performed learning using two learning rates, α_slow and α_fast, where the former matches the learning rate used in [40] and the latter is 10 times faster. Each bar corresponds to 5 runs of the whole test set (using different random seeds). Error bars correspond to three-sigma in each direction.

By comparing the blue bars to the yellow bars in Fig 10, it becomes evident that a faster learning rate is detrimental to our SNN. With a slower learning rate, the 'noise' introduced from process variation has less impact on the classification rate (synapses get more opportunities to learn input patterns accurately). Another clear trend seen in the results from Fig. 10 is that increasing the number of neurons leads to much higher classification rates. By replicating the experiments reported in [40], an average classification rate of 91.5% is achieved (dark green bar). When the same network structure is used, but the weights are mapped to our binary-weighted synapses, the average classification rates drops to 90.64% (light green bar). For the worst case of the 5 runs performed, our proposed SNN classified 947 digits incorrectly.

V. Conclusion

We presented a novel probabilistic synapse which is integrated with a fully-digital neuron to form a SNN. The design decisions and optimizations are discussed, from device-level to system-level. The synapse's logic-in-memory type of architecture requires no separate memory access mechanism. Instead, the proposed cross-coupled latch-based access mechanism performs memory access (and computation) in analog domain with significantly less power while generating a digital output signal which can be seamlessly connected to neurons of the network.

The integration of analog and digital domains exploits the low-power nature from the S-MTJ and the ease of implementation of a digital system. Moreover, the non-volatility of an MTJ makes it an interesting choice for a low-power normally off device as the weights are persistent. An STDP-based hardware friendly learning scheme is also discussed in regards to the proposed system. Finally, the proposed MTJ-based neural network shows remarkably promising power and area efficiency. Its functionality is demonstrated by unsupervised training and classification using the MNIST database of handwritten digits.

Acknowledgment

This work was supported by the National Science Foundation under contract CCF-1714334.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks," In *Advances in neural information processing systems,* pp. 1097-1105, 2012.

[2] K. He, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE international conference on computer vision,* pp. 1026-1034, 2015.

[3] J. Hennessy, and D. A. Patterson *"Computer architecture: a quantitative approach",* Amsterdam, The Netherlands: Elsevier, 2012.

[4] A. Nowatzyk, F. Pong, and A. Saulsbury, "Missing the Memory Wall: The Case for Processor/Memory Integration," *23rd Annual International Symposium on Computer Architecture (ISCA),* pp. 90-90, 1996.

[5] J. Warnock, "Circuit design challenges at the 14nm technology node," *2011 Proceedings of the 48th Design Automation Conference,* pp. 464-467, 2011.

[6] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. National Academy of Sciences. USA,* vol. 79, pp. 2554-2558, 1982.

[7] P. A. Merolla et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science,* vol. 345, no. 6197, pp. 668-673, 2014.

[8] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks* 10.9, pp. 1659-1671, 1997.

[9] V. Jilles, *"Spiking neural networks, an introduction,"* Institute for Information and Computing Sciences, Utrecht University Technical Report, 2002.

[10] R. Van Rullen, and S. J. Thorpe, "Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex," *Neural computation,* vol. 13.6, pp 1255-1283, 2001.

[11] S. J. Thorpe, "Spike arrival times: A highly efficient coding scheme for neural network," *Parallel processing in neural systems,* pp 91-94, 1990.

[12] N. Kasabov, "To spike or not to spike: A probabilistic spiking neuron model," *Neural Networks,* vol. 23.1, pp 16-19, 2010.

[13] J. S Seo, et al., "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," *2011 IEEE Custom Integrated Circuits Conference (CICC),* pp. 1-4, 2011.

[14] P. Knag, J. K. Kim, T. Chen, Z. Zhang, "A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding," *IEEE Journal of Solid-State Circuits,* 50(4), pp. 1070-1079, 2015.

[15] J. K Kim, P. Knag, T. Chen, and Z. Zhang, "A 640m pixel/s 3.65 mw sparse event-driven neuromorphic object recognition processor with on-chip learning," *2015 Symposium on VLSI Circuits,* pp. C50-C51, 2015.

[16] D. Kuzum, Y. Shimeng, and HS Philip Wong, "Synaptic electronics: materials, devices and applications," *Nanotechnology,* vol. 24.38, pp. 382001, 2013.

[17] S. Park, et al. "RRAM-based synapse for neuromorphic system with pattern recognition function," *2012 International Electron Devices Meeting (IEDM),* pp. 10.2.1-10.2.4, 2012.

[18] D. Querlioz, et al. "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE Transactions on Nanotechnology,* vol. 12.3, pp 288-295, 2013.

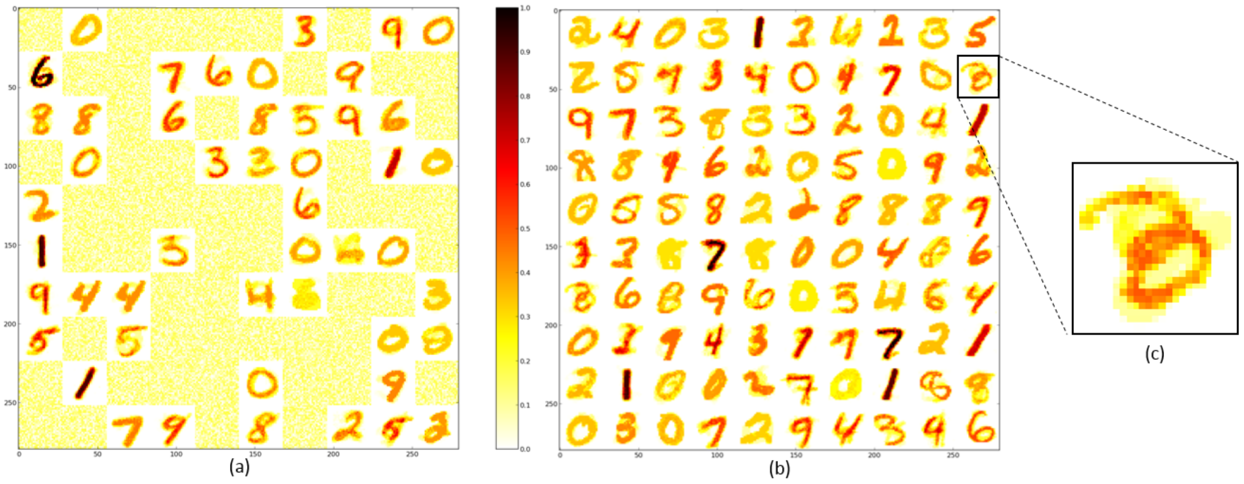

Fig. 9: (a) Excitatory synaptic weights after 6000 training samples are provided. (b) Excitatory synaptic weights after 12000 training samples are provided. The digits on one side do not correspond to the other side, as the stochastic nature of the network assigns different digits to different neurons. (c) Blurry pattern that resembles the digit 3.

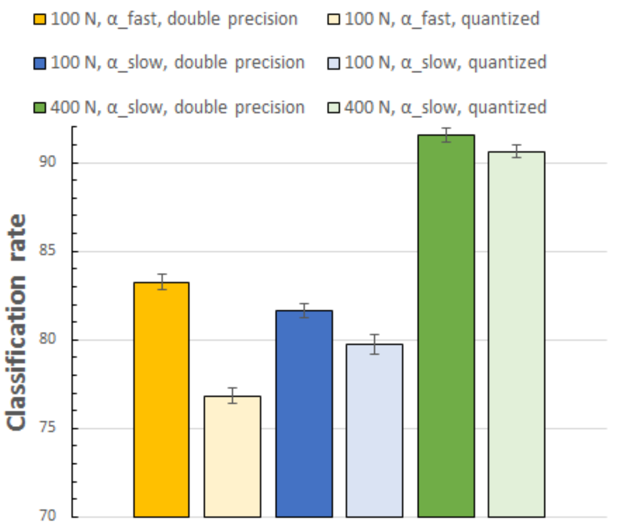

Fig. 10: Classification rates for different learning rates and network sizes, considering double-precision weights and quantized weights.

[19] M. Prezioso, et al., "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature,* vol. 521(7550), pp. 61, 2015.

[20] J. G. Zhu, and C. Park, "Magnetic Tunnel Junctions," *Materials Today,* vol. 9, pp 36-45, 2006

[21] Z. Sun, "Spin angular momentum transfer in current-perpendicular nanomagnetic junctions," in *IBM Journal of Research and Development,* vol. 50, no. 1, pp. 81-100, 2006.

[22] S. Bhuin, A. K. Biswas and L. Pileggi, "Strained MTJs with latch-based sensing for stochastic computing," *2017 IEEE 17th International Conference on Nanotechnology (IEEE-NANO),* pp. 1027-1030, 2017.

[23] S. Bhuin, et al., "A self-calibrating sense amplifier for a true random number generator using hybrid FinFET-straintronic MTJ," *2017 IEEE/ACM International Symposium on Nanoscale Architectures,* pp. 147-152, 2017.

[24] M. Barangi, J. S. Chang, and P. Mazumder, "Straintronics-Based True Random Number Generator for High-Speed and Energy-Limited Applications," *IEEE Transactions on Magnetics,* vol. 52, pp. 1-9, 2016.

[25] A. K. Biswas, S. Bandyopadhyay, and J. Atulasimha, "Complete magnetization reversal in a magnetostrictive nanomagnet with voltage-generated stress: A reliable energy-efficient non-volatile magneto-elastic memory", *Appl. Phys. Lett.,* vol. 105, pp. 072408, 2014.

[26] J. Alspector, et al., "A VLSI-efficient technique for generating multiple uncorrelated noise sources and its application to stochastic neural networks," *IEEE Transactions on circuits and systems,* vol. 38.1, pp 109-123, 1991.

[27] V. Fischer, and M. Drutarovský, "True random number generator embedded in reconfigurable hardware," *Cryptographic Hardware and Embedded Systems - CHES 2002 Lecture Notes in Computer Science,* vol. 2523, pp. 415-430, 2003.

[28] D. B. Thomas, H. Lee, and L. Wayne, "A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation," *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays,* pp. 63-72, 2009.

[29] K. L. Wang, J. G. Alzate, and P. K. Amiri, "Low-power non-volatile spintronic memory: STT-RAM and beyond," *Journal of Physics D: Applied Physics,* vol. 46.7, 2013.

[30] A. K. Biswas, H. Ahmed, J. Atulasimha, S. Bandyopadhyay, "Experimental Demonstration of Complete 180° Reversal of Magnetization in Isolated Co Nanomagnets on a PMN-PT Substrate with Voltage Generated Strain," *Nano Lett.,* vol. 17, 6, pp. 3478-3484 , 2017.

[31] D. H Goldberg, G. Cauwenberghs, and A. G. Andreou. "Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons," *Neural Networks,* vol. 14.6, pp. 781-793, 2001.

[32] W. Zhao, et al. "High speed, high stability and low power sensing amplifier for MTJ/CMOS hybrid logic circuits," *IEEE Transactions on Magnetics,* vol. 45.10, pp. 3784-3787, 2009.

[33] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE transactions on neural networks,* vol. 15.5, pp. 1063-1070, 2014.

[34] R. Kempter, G. Wulfram, and J. Leo Van Hemmen, "Hebbian learning and spiking neuron," *Physical Review,* vol. E 59.4, pp. 4498, 1999.

[35] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience* 3.9, pp 919-926, 2000.

[36] W. Gerstner, et al., "A neuronal learning rule for sub-millisecond temporal coding," *Nature,* vol. 383(6595), pp. 76, 1996.

[37] D. Apalkov, B. Dieny and J. M. Slaughter, "Magnetoresistive Random Access Memory," *Proceedings of the IEEE,* vol. 104, no. 10, pp. 1796-1830, 2016.

[38] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86.11, 1998.

[39] D. F. M. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in Python," Frontiers in Neuroinformatics, vol. 2, 2008.

[40] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," Frontiers in Computational Neuroscience, vol. 9, 2015.