INVITED- Cross-Layer Approximations for Neuromorphic Computing: From Devices to Circuits and Systems

Priyadarshini Panda, Abhronil Sengupta, Syed Shakib Sarwar, Gopalakrishnan Srinivasan, Swagath Venkataramani, Anand Raghunathan and Kaushik Roy
School of Electrical and Computer Engineering, Purdue University
{pandap, asengup, sarwar, srinivg, venkata0, raghunathan, kaushik}@purdue.edu

ABSTRACT

Neuromorphic algorithms are being increasingly deployed across the entire computing spectrum from data centers to mobile and wearable devices to solve problems involving recognition, analytics, search and inference. For example, large-scale artificial neural networks (popularly called deep learning) now represent the state-of-the art in a wide and ever-increasing range of video/image/audio/text recognition problems. However, the growth in data sets and network complexities have led to deep learning becoming one of the most challenging workloads across the computing spectrum. We posit that approximate computing can play a key role in the quest for energy-efficient neuromorphic systems. We show how the principles of approximate computing can be applied to the design of neuromorphic systems at various layers of the computing stack. At the algorithm level, we present techniques to significantly scale down the computational requirements of a neural network with minimal impact on its accuracy. At the circuit level, we show how approximate logic and memory can be used to implement neurons and synapses in an energy-efficient manner, while still meeting accuracy requirements. A fundamental limitation to the efficiency of neuromorphic computing in traditional implementations (software and custom hardware alike) is the mismatch between neuromorphic algorithms and the underlying computing models such as von Neumann architecture and Boolean logic. To overcome this limitation, we describe how emerging spintronic devices can offer highly efficient, approximate realization of the building blocks of neuromorphic computing systems.

1. INTRODUCTION

The explosion in various forms of digital data has led to a great demand for computing platforms to perform tasks such as recognition, analytics, and inference. The brain's ability to perform many such tasks with great energy efficiency has driven interest in the field of neuromorphic computing, which broadly refers to the use of algorithms or hardware inspired by the brain. In fact, large-scale brain-inspired neural networks such as Deep Learning Nets [1,2], Hierarchical Temporal Memory [3] etc. have been employed in many real-world applications such as image search [4] and speech recognition [5] among others. However, these applications are highly compute intensive and their implementations on software (on multi-cores or GPGPUs) as well as custom hardware (analog or digital CMOS) expend significant amounts of energy [7]. With energy-efficiency becoming a primary concern across the computing spectrum, energy-efficient realization of neuromorphic applications is of great interest.

DAC'16, June 05-09, 2016, Austin, TX, USA
 ISBN 978-1-4503-4236-0/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2897937.2905009

Neuromorphic applications invariably exhibit very high levels of "intrinsic resilience" to approximations; in other words, they have an innate ability to produce acceptable results even when some of the underlying computations are performed in an imperfect or approximate manner [6]. The intrinsic resilience of such applications can be attributed to several factors [8]: (i) The algorithms process input datasets that contain significantly large redundancies and noise that impart them with robustness to noise in the input data as well as approximate computations, (ii) The algorithms employ computational patterns such as iterative refinement that can heal or correct the errors (introduced by approximations) by averaging them out in successive iterations.

In [6], the authors have quantitatively established that 83% of the runtime across a benchmark suite of 12 recognition and mining applications is spent in error resilient computations. Thus, there is significant scope for power and performance optimization by forsaking the requirement for exact (Boolean) equivalence between implementation and specification. This opportunity is not unique to neuromorphic applications; other application domains such as Digital Signal Processing (DSP), multimedia (image/video/audio), graphics, and wireless communications also demonstrate intrinsic resilience. Several approximate design techniques for both software and hardware implementations of applications from these have been proposed [9-12]. In DSP for example, filter designs can be approximated by optimizing the taps, coefficients, and precision based on specifications [13]. Recent advances in approximate computing have been highlighted in [48]. It is worth mentioning that neuromorphic applications are unique due to the ability to consider the effect of approximations while training, or to re-train the network to mitigate the effects of approximations. This allows us to introduce approximations at every layer of the stack with minimal loss in accuracy, thus achieving large improvements in energy consumption.

Over the years, the field of neuromorphic computing have broadened from neural algorithms to hardware architectures and circuits that loosely mimic various aspects of cortical information processing. We believe that approximate computing can play a central role in these efforts as well. For example, IBM's digital CMOS neuromorphic chip called TrueNorth integrated memory and computation into a single unit [14]. The choice to limit synaptic precision enables the chip to use only 32 CMOS transistors to realize a synaptic weight, improving efficiency. Recent research suggests that emerging post-CMOS devices can approximately realize neural and synaptic functionality with drastic reduction in energy consumption [35, 44].

In this work, we outline a range of approximate design techniques that we have developed at various layers of the computing stack

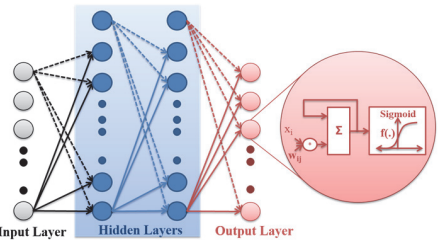

Fig. 1 Feedforward ANN with input, hidden and output layers

from algorithms to circuits to devices for energy-efficient realization of neural networks. Note that this paper is part of DAC 2016 Special Session: "Cross-Layer Approximate Computing: Challenges and Solutions". Other papers in this special session are [45-47].

2. NEURAL NETWORKS: BACKGROUND

Artificial Neural Networks (ANNs) are interconnected networks of artificial neurons and synapses that are loosely modeled after the information processing structures in the human brain. The neurons are arranged in different layers viz., input, one or more hidden and output layers. The network shown in Fig. 1 is a feedforward ANN with fully connected interconnections from one layer to the next. Every neuron in the ANN sums the product of the incoming inputs and the connecting weights. Subsequently, the summed result is fed to an activation or thresholding function to obtain the final output. The activation function can be hard-limiting (e.g. step function) or soft-limiting (e.g. sigmoid or rectified linear functions). The weights of an ANN are the adjustable parameters that can be tuned during training to obtain desired outputs for specific inputs.

ANNs are commonly trained using the backpropagation algorithm [2] with supervision from a training dataset. The central idea of training is to iteratively update the weights such that the output error is minimized. A trained ANN stores information in a highly distributed manner, enabling it to tolerate errors in the basic neuronal computations and weight perturbations. This error resilience can be exploited to approximate the basic computational elements of ANNs – neurons and synapses – and achieve energy efficiency.

While large scale deep ANNs have achieved considerable success in a range of applications, there is growing interest in a new, more biologically realistic generation of NNs: Spiking Neural Networks (SNNs), which represent neuronal outputs by means of spikes, thereby performing computation in an event-driven fashion. The asynchronous occurrence of individual spikes also forms the basis of communication in SNNs. A range of neuronal and synaptic models with varying levels of biological fidelity have been proposed in SNNs, of which popular examples are the Leaky-Integrate-Fire (LIF) neuron model [40] and Spike Timing Dependent Plasticity (STDP) model for synaptic learning [43]. SNNs focus their computation and communication effort only on the active parts of the network as determined by spiking events, thereby offering the potential for lower power consumption. Recent research has shown that individual spintronic devices can approximate the building blocks of SNNs, resulting in highly compact and energy-efficient implementations.

Having discussed the background of NNs and their basic computational elements, we will now discuss the approximations that can be introduced at various layers of the computational stack to realize these networks in an energy-efficient manner.

3. ALGORITHMIC LEVEL

At the top layer of the computing stack, we can introduce approximations based on domain-specific insights to obtain computational efficiency and hence, energy improvement. Contemporary NN development techniques focus primarily on accuracy, often resulting in networks with significantly higher computational requirements than necessary. For instance, Deep Learning Networks, that have become popular due to their state-of-the-art performance on several machine learning problems [26], typically spend the same amount of computational effort on every input, regardless of the fact that in practice, inputs vary greatly in their inherent difficulty. Based on these observations, we propose algorithmic techniques to transform deep learning networks to reduce their computational requirements.

3.1 Staged Networks for Scalable Computational Effort

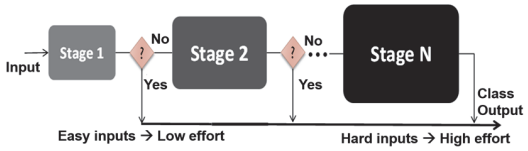

Fig. 2 Scalable effort classifier as a sequence of decision models which grow progressively complex [27]

For many computer vision applications, all inputs may not have equal significance. Consider the simple example of recognizing a person from two images: one where the person is standing against a plain blue backdrop and other where the person is in the midst of a crowd. Clearly, the latter one takes more time and effort. Yet, state-of-the-art neural networks including Deep Learning Convolutional Neural Networks (DLNNs) are designed to be fixed-effort systems i.e., the same computational structure is used to evaluate all inputs irrespective of their difficulty, leading to significant inefficiency. To address this limitation, the concept of staged networks is proposed, as shown in Fig. 2 [27]. The key idea in staged networks is to build a sequence of neural network stages of increasing complexity (and accuracy). The input is fed into the first stage; each stage is equipped to gauge its confidence in classifying the input and passes the input to the next stage, if required. Thus, easy inputs are classified by the initial stages with very low effort, while subsequent stages are invoked only for more challenging inputs. The average reduction in compute complexity is determined by the fraction of inputs classified in each stage and their relative complexity with respect to the original neural network. Each stage of the scalable network is built with an ensemble of biased classifiers, where each biased classifier is trained to detect a single class more accurately [27].

Given any classification algorithm, different models are learnt using the same algorithm and training data. These models are then connected in a sequence such that the initial stages are computationally efficient but have lower classification accuracies, while the later ones have both higher complexities and accuracies as illustrated in Fig. 2. Further, each stage in the cascade is also designed to implicitly assess the "hardness" of the input. Experiments on various benchmark datasets reveals that such approaches can yield 2-3X reduction in average operations per input while maintaining a competitive classification accuracy with respect to the baseline network [27].

3.2 Conditional Deep Learning

Let us investigate the application of fundamental variability in the difficulty of input instances on a deep learning convolutional framework. It is well-known that the convolutional layers (CNN layers) of a DLNN, interpreted as visual layers, learn a hierarchy of features which transition from general (similar to Gabor filters and color blobs [28]) to specific, as we go deeper into the network [29]. We can utilize the generic-to-specific transition in the learnt features

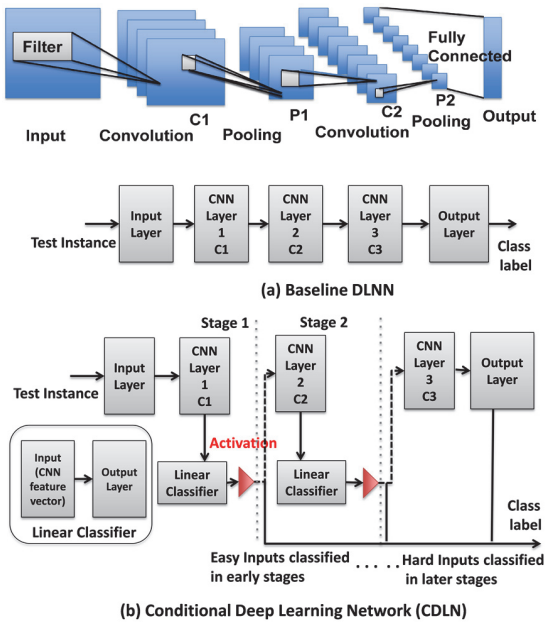

Fig. 3 Conditional Deep Learning Methodology with additional output layers [30]

of the CNN layers to discriminate between easy and hard inputs in a dataset and implement Conditional Deep Learning (CDL), shown in Fig. 3 [30]. One way of achieving such a goal is to add a linear network of output neurons for each convolutional layer and monitor the output of the linear network to conditionally activate the deeper layers. In other words, instead of drawing the output from the complex DLNN model for all input instances, one can draw the outputs from the linear networks.

Depending upon the output of the linear classifiers in a given stage, the following stage of the CDLN is enabled. As we go deeper into the network, the decision boundary model for each stage in the CDLN becomes progressively non-linear. Thus, class labels are produced at stage 1 for easy inputs and latter stages for hard ones. During test time, input instance is passed through each stage to produce a class label. Evaluation of the CDL methodology on the MNIST dataset demonstrates 1.91x reduction in average operations per input [30]. In addition to energy efficiency, we also observe that the CDL network outperforms the baseline DLNN in terms of classification accuracy (97.5% for DLNN and for 98.9% CDL). This can be attributed to the fact that the linear networks being small scale with few neurons and synapses can be trained rapidly and easily to achieve better least mean square error as compared to the baseline DLNN. Such architecture adds the intelligence of a deep neural network to resource constrained devices and would be suitable for neuromorphic hardware for real-world applications.

3.3 Approximate Neural Network

Having accounted for input variability, we explored a new direction to improve the efficiency of DLNNs that leverages backpropagation training to maximize the energy-benefits from approximating a neural network [25]. Two key properties of neuromorphic systems motivate our work: a) DLNNs are used in applications like recognition, search etc. where imperfect results are acceptable, and b) inherent resiliency to inexactness in the constituent computations in DLNNs. Inspired from this, we proposed a method to transform any given DLNN into Approximate NN (AxNN) where certain neurons are replaced with approximate versions that are more energy-efficient.

A key question in approximate computing is which computations to approximate, and by how much. The judicious selection of approximations is critical to maximizing energy-efficiency while maintaining acceptable output quality. In neuromorphic systems, we notice that backpropagation provides a sensitivity of the network outputs to each neuron in the network. We use the backpropagation training to *characterize* the importance of each neuron and identify those that impact output quality the least. Then, AxNN is created by replacing the less significant neurons with *approximate* ones. We utilize precision scaling, a popular approximate design technique, and modulate the precision value of the inputs and the weights to obtain the best energy-accuracy tradeoff. Once the AxNN is formed, we relearn the weights in the approximated network with incremental *retraining* to mitigate the effect of approximations on output quality. In [25], we have developed a systematic methodology to create AxNNs by iterating the *characterize*, *approximate* and *retraining* steps mentioned above in a quality-constrained loop. We evaluated the proposed approach by constructing AxNNs for 6 recognition applications. Our results demonstrate up to 1.92x energy benefits for virtually no loss (<0.5 %) in accuracy, and even higher improvements up to 2.3x for a tolerable (up to 7.5%) loss in accuracy.

4. CIRCUIT LEVEL

When it comes to designing approximate circuits, two broad approaches have been followed. The first approach is based on designing circuits in over-scaled conditions (voltage) to achieve large improvements in energy, albeit with some (controlled) accuracy degradation. However, over-scaling can have a lasting impact on the MSBs resulting in unacceptable performance loss [15]. The second approach approximates the fundamental logic functions so as to reduce the hardware complexity (deviation from precise specification with few transistors or gates [16-20]) thereby achieving area and energy benefits. In [17], for instance, we used an approximate full adder (inexact truth table for a few inputs) to evaluate image compression algorithms with up to 60%/37% power/area savings with insignificant loss in output quality. We base our techniques on the above two approaches for approximating the functional units of a NN pertaining to neuromorphic recognition applications as described in subsequent sections.

4.1 Approximate Multiplier-less Neuron

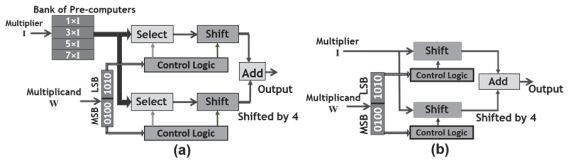

Fig. 4. (a) 8 bit- 4 alphabet ASM (b) 1 alphabet {1} ASM or Multiplier less neuron [21]

It is well-known that multiplication operation in the hardware implementation of NNs accounts for a significant portion of power dissipation. To reduce this energy consumption, we proposed an approximate Alphabet Set Multiplier (ASM) [21] that utilizes the notion of computation sharing [22] to implement a digital NN. In ASM, a conventional multiplication is replaced by simplified shift and add operation on "alphabets". Alphabets are 4-bit values that can be used to represent any digital binary number. We use the alphabets to decompose a multiplication operation into smaller components. For instance, if we want to multiply '*I*' with multiplicand '*W*' (=00110001_2), the multiplication can be expressed as $2^4(0011_2)\times I + 2^1(0001_2)\times I$ where "0011_2={3}" and "0001_2={1}" are the alphabets used to represent '*W*'. With this decomposition, the final product (W×I) can, thus, be obtained with simple bit-shift and

addition operations of lower order product terms ($1\times I$, $3\times I$). These lower order product terms are computed using a pre-computer bank in an ASM. Fig. 4(a) illustrates an 8-bit 4-alphabet ASM. In addition to the shift and add units, the ASM contains a select unit that selects the lower order product term from the pre-computer bank based on the multiplicand '*W*'. The control logic is used to implement different combinations of the above operations.

To ensure correct multiplication result with 4-bit alphabets, 8 alphabets {1,3,5,7,9,11,13,15} are generally required in an ASM. It is evident that the power dissipation of the pre-computer unit (that accounts for a large portion of ASM's energy consumption) is proportional to the number of alphabets. However, exploiting the error resilience of NNs, we reduce the number of alphabets in the ASM to achieve lower routing complexity and power dissipation at a tolerable accuracy degradation. The reduced alphabet set, {1,3,5,7} for instance, cannot generate all possible combinations of products and would lead to a decline in accuracy. To alleviate this problem, we retrained the network with imposed constrains (i.e. reduced alphabet set in ASM).

Our initial evaluation of approximate (with different combinations of reduced alphabet set) ASM neurons on MNIST [2] dataset demonstrated that the accuracy degradation was within a marginal limit of ~0.5% even when the reduced alphabet set contained only 1 alphabet: {1}. A {1}-alphabet ASM is shown in Fig. 4(b). It is clearly seen that the pre-computer bank and alphabet select unit is no longer required. . Elimination of the above units provides significant power savings with faster and compact representation for multiplication operation. Our experiments on 4 benchmark datasets in [21] using the multiplier-less neuron (Fig. 4(b)), demonstrate ~35% and ~60% reduction in power consumption, and ~37% and ~62% reduction in area, respectively, for 8 and 12 bit neurons for a tolerable loss (up to 2.83%) in accuracy.

4.2 Significance-driven Hybrid 8T-6T SRAM for On-Chip Synaptic Storage

Now, we discuss the approximations that can be applied to the synaptic units of an NN to design an energy-efficient on-chip storage for the synaptic weights. Note that the number of synapses is orders of magnitude larger than the number of neurons. Hence, an on-chip synaptic memory designed using a conventional 6T SRAM as shown in Fig. 5(a), contributes substantially to the power consumption of a typical ASIC implementation of ANNs. Now, let us consider supply voltage scaling to achieve energy efficiency. The 6T SRAM is prone to bit-cell failures at scaled voltages due to the detrimental impact of process parameter variations [41]. However, ANNs being error resilient applications, tolerate moderate perturbations in the stored synaptic weights. Our analysis on the MNIST [3] dataset indicates that the voltage can be scaled by up to 200mV from the nominal operating voltage (950mV) with minimal loss (less than 0.5%) in the classification accuracy (22nm predictive technology [42]). However, scaling the voltage beyond 200mV causes a substantial degradation in accuracy due to increased probability of failures in the MSBs of the synaptic weights.

We propose a significance driven hybrid 8T-6T SRAM (Fig. 5(b)) wherein the sensitive MSBs are stored in 8T bit-cells while the relatively resilient LSBs are stored in 6T bit-cells [23]. The 8T bit-cells provide reliable operation under scaled supply voltage due to the presence of independently optimized read and write paths. Furthermore, the hybrid 8T-6T arrays can be effectively laid out in a single row, and hence does not incur any other overhead aside from an obvious area penalty due to an increase in the transistor count [24]. The stable operation of 8T bit-cells enables the voltage of the hybrid array to be scaled by another 100mV. This provides a 29% improvement in the memory access and leakage power

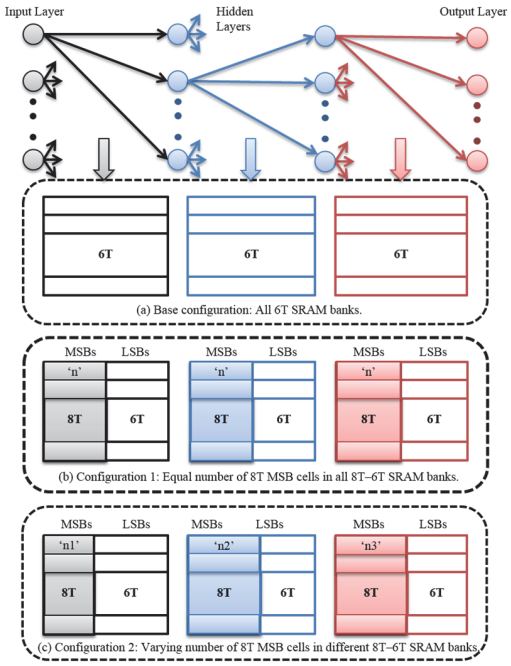

Fig. 5. Synaptic memory configurations under study. (a) All 6T SRAM. (b) Significance driven hybrid 8T-6T SRAM. (c) Synaptic-sensitivity driven hybrid memory architecture.

consumption, albeit with a 13.75% area penalty. Considering the error resiliency of the application, a synaptic-sensitivity driven hybrid memory architecture (Fig. 5(c)) consisting of multiple 8T-6T SRAM banks, each of which stores synapses carrying a definite significance, can be used to further minimize the area overhead. The proposed architecture determines the number of 8T MSB cells for the synapses interconnecting different layers of the ANN based on their sensitivity. For instance, the synapses fanning out of the input layer and the first hidden layer that extracts low-level features from the input dataset are significant relative to those interconnecting the central hidden layers. The resilient synapses have relatively fewer MSBs stored in 8T bit-cells in comparison to the ones that are deemed significant. Our analysis on the MNIST dataset shows that the synaptic-sensitivity driven architecture provides a 30.91% reduction in the memory access power with a 10.41% area overhead, for less than 1% loss in the classification accuracy.

5. EMERGING DEVICES FOR NEUROMORHIC COMPUTING

The discussions so far, mainly the section on approximations at the algorithm level, have been more or less, device agnostic. The focus on the circuit section has been on scaled CMOS devices and how voltage scaling can lead to proper trade-offs between accuracy and energy consumption. In this section we will provide a discussion on emerging devices, especially spintronic devices that have the potential to approximately mimic the neuronal and the synaptic functions. However, as mentioned, the functions that these devices implement may not exactly match the standard activation function (such as the sigmoid function) of an ANN (Fig. 1), or may lead to reduced bit-precision in synaptic arrays. For other computing models such as SNNs, the proposed spin neurons may not directly implement the Leaky Integrate Fire (LIF) model [40] that is commonly used. As we will observe later, the LIF function can be well-approximated by the magnetization dynamics of a nano-magnet. To fully exploit the potential of such low-power devices, the use of a cross-layer approach to design (from algorithms to

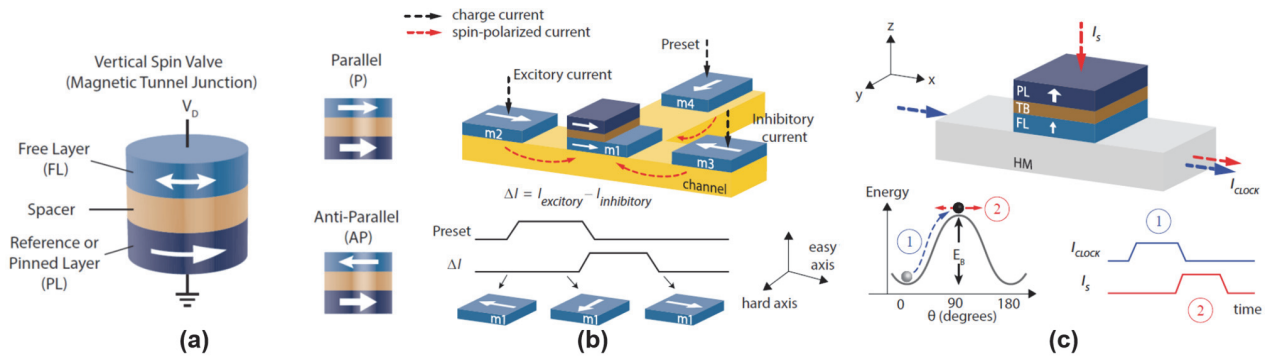

Fig. 6. Spintronic devices emulating the step-transfer function of an Artificial Neuron: (a) Magnetic Tunnel Junction, (b) Lateral Spin Valve based structure, (c) Spin-orbit torque based structure

devices) is essential. In this section, we will briefly describe the functionalities of the proposed spin-transfer torque devices and their suitability and application for neuromorphic computing.

Before discussing the manner in which neural and synaptic operations can be approximated by emerging spintronic devices, let us first illustrate a few basic concepts regarding their principle of operation. A nano-magnet is usually characterized by two stable magnetization directions, which are referred to as the "easy axis" for the magnet. The direction is dictated by the anisotropy energy of the magnet. For example, for a magnet with Perpendicular Magnetic Anisotropy (PMA), the stable magnetization directions are out-of-plane, namely "up-spin" and "down-spin" directions. For a mono-domain magnet, the magnetization direction of the magnet (free layer) can be sensed by using a Magnetic Tunnel Junction (MTJ) structure where the free magnet is separated from another magnet with fixed magnetization (referred to as the "Pinned" layer) by a tunneling oxide barrier. The MTJ exhibits two extreme resistive states, namely the low resistive "Parallel" (P) state (both magnets oriented in the same magnetization direction) and the high-resistive "Anti-Parallel" (AP) state (magnets oriented in the opposite magnetization direction). The magnetization of the free layer can be switched using a spin polarized current (spin-transfer torque – STT switching) or by recently discovered energy-efficient techniques like spin-orbit torque switching enabled by current flowing through a heavy metal underlayer [37-38]. In addition to mono-domain magnets, magnets with a domain wall separating two oppositely oriented magnetic domains have also been fabricated. We will refer to such a magnet as a Domain-Wall Magnet (DWM) for the rest of this text. The domain wall location in such a magnet can be displaced by an input current flow through the magnet.

Let us first consider the neural thresholding operation in an Artificial Neural Network (ANN). The step transfer function of a neuron in an ANN, i.e. neuron switching to one state or another depending on the sign of the resultant synaptic input can be approximated by the STT switching of a magnet, as discussed previously. Depending on the direction of the input spin current, the MTJ will switch either to the P or AP configuration, provided the current magnitude is greater than a particular threshold value [31]. In addition to such standard STT switching, alternative schemes have been employed to orient the magnetization direction of the magnet along an unstable direction (referred to as the "hard-axis") and subsequently switching it by a very small input synaptic current. Such "hard-axis" orientation of nanomagnets for neural thresholding operations have been explored in Lateral-Spin Valve (LSV) based structures [32, 33] and spin-orbit torque induced PMA magnet switching [34]. Due to orientation of the magnet along the "hard-axis", the resultant synaptic current requirement to switch the magnet deterministically to either one of the two stable states reduces drastically, thereby approximating the step transfer function of the neuron to a greater degree of precision. Fig. 6 illustrates the corresponding spintronic device structures that can be utilized for the neural step function implementation.

Non-step neuron transfer functions, that can potentially encode a greater degree of information than the step transfer function neurons, can be emulated by DWM nanostrips where the magnet is a part of an MTJ structure. Depending on the position of DW in the ferromagnet, the resistance of the device will vary due to variation in the relative proportion of P and AP domains in the MTJ. Further, as shown in Fig. 7(a), the DWM can be interfaced with a heavy metal (HM) underlayer to achieve energy-efficient spin-orbit torque driven DW motion [39]. Such a device can be interfaced with a "Reference" MTJ (whose orientation is always fixed in the AP direction) to form a resistive divider network, which in turn drives the gate of an output transistor. The domain wall location in the device determines the gate voltage, and in turn, the current provided by the output transistor (Fig. 7(b)). By properly biasing the output transistor, it can be shown that the relationship between the output current provided by the transistor and the input synaptic current flowing through the device bears an approximate linear relation [35].

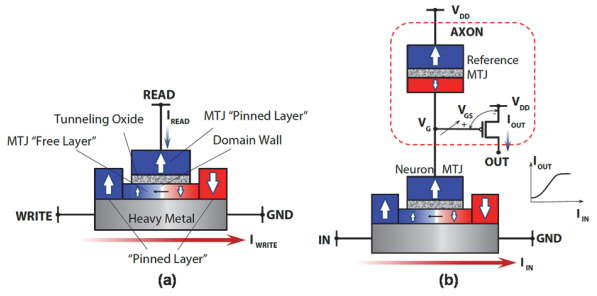

Fig. 7. (a) DWM based device structure for neuromorphic applications, (b) Operation of the device as a neuron.

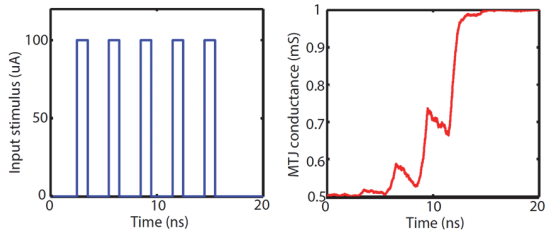

Fig. 8. The MTJ conductance on the receipt of an input stimulus and leaks in the absence of the stimulus.

In addition to neuronal operations, such a device can be operated as a multi-bit synapse where the synaptic weight is encoded in the device resistance by programming the domain wall position [35, 43]. The number of programming levels offered by such spintronic devices will be limited by the dimensions of the device along with associated thermal noise and imprecision in the programming levels. However, as mentioned previously, most of these neuromorphic algorithms are error-resilient and require synaptic weights with 4-5 bit resolution. Hence, synaptic functionalities can be emulated by such emerging technologies to a fair degree of precision and simultaneously achieve a reasonable accuracy in complex recognition problems. Such spintronic synapses can potentially be interfaced with spintronic neurons, thereby leading to the implementation of an all-spin ANN design [35].

In addition to the second generation of neural networks, namely ANNs, the magnetization dynamics of a nano-magnet bear striking resemblance to the computing model for SNNs. The magnetization of the magnet, which in turn, dictates the MTJ resistance increases on the receipt of an input current pulse. However, in the absence of a pulse it starts reducing back to its initial state (Fig. 8) [36]. Such a functionality can be mapped to the leak and integrate properties of a biological spiking neuron. In addition, thermal noise provides a degree of randomness to the spiking behavior, thereby providing a direct mapping to the stochastic firing nature of biological neurons in the cortex.

Device-circuit-algorithm co-design reveals that such spintronic neural computing platforms where the neural and synaptic functionalities are approximated by the underlying device physics can achieve competitive classification accuracy in a large number of complex recognition problems [35]. Further, the ultra-low current mode operation of magneto-metallic spintronic neurons enable the ultra-low voltage operation of spintronic crossbar arrays, resulting in ~100x improvement in energy consumption in comparison to a corresponding 45nm digital CMOS implementation [35].

6. CONCLUSION

Neuromorphic computing has found widespread use across computing platforms to implement cognitive applications. The abundant growth in data sets and neural network complexities has led to the quest for novel techniques to implement neuromorphic platforms in an energy-efficient manner. Neural algorithms exhibit inherent error resilience that has been exploited in several ways to develop approximations at various layers of the computing stack to obtain energy benefits. While many advances have been made in this area, much remains to be explored and many challenges need to be addressed. These include the development of models that effectively use feedback connections as in the brain to perform cognition efficiently, integration of error resiliency at various layers of the stack leading to a cohesive framework for evaluation of proposed techniques on real-end systems. Addressing these key issues is imperative for true energy-efficient realization of neuromorphic systems.

ACKNOWLEDGEMENTS

This work was supported in part by C-SPIN, one of the six centers of StarNnet, a Semiconductor Research Corporation Program, sponsored by MARCO and DARPA, by the Semiconductor Research Corporation, the National Science Foundation, Intel Corporation and by the National Security Science and Engineering Faculty Fellowship.

7. REFERENCES

[1] A. Krizhevsky, et. al "Imagenet classification with deep convolutional neural networks.",In Proc. NIPS, 2012.
[2] Y. LeCun, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 1998.
[3] J. Hawkins et. al Hierarchical temporal memory: Concepts, theory and terminology. Technical report, Numenta, 2006.
[4] "Improving photo search: A step across the semantic gap", 2009.
[5] J. Dean, et al. "Large scale distributed deep networks." In Proc. NIPS, 2012.
[6] V. K. Chippa et. al Analysis and characterization of inherent application resilience for approximate computing. In Proc. DAC, 2013.
[7] A. Sandberg, "Energetics of the brain and AI.", arXiV:1602.04019, 2016.
[8] V. K. Chippa et. al. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In Proc. DAC, 2010.
[9] N. Shanbhag, "Reliable and energy-efficient digital signal processing," In Proc. DAC 2002.
[10] D. Mohapatra et. al "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," In Proc. ISLPED 2009.
[11] S. Narayanan et. al. Scalable stochastic processors. In Proc. DATE, 2010.
[12] J. Han et. al. Approximate computing: An emerging paradigm for energy efficient design. In Proc. ETS, 2013.
[13] V. Madisetti. Digital Signal Processing 2nd Ed. CRC Press, 2008.
[14] P. A. Merolla et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface." Science, 2014.
[15] V. K. Chippa et. al. Scalable effort hardware design. TVLSI, 2014.
[16] G. Karakonstantis et. al. An optimal algorithm for low power multiplier less fir filter design using Chebychev criterion. ICASSP, 2007.
[17] V. Gupta et. al. IMPACT: imprecise adders for low-power approximate computing. In Proc. ISLPED, 2011.
[18] S. Venkataramani et. al. SALSA: systematic logic synthesis of approximate circuits. In Proc. DAC, 2012.
[19] S. Venkataramani et. al. Substitute-and-simplify:A unified design paradigm for approximate and quality configurable circuits. In Proc. DATE, 2013.
[20] A. Ranjan et. al. ASLAN: synthesis of approximate sequential circuits. In Proc. DATE, 2014.
[21] S. Sarwar et. al Multiplier-less Artificial Neurons Exploiting Error Resiliency for Energy-Efficient Neural Computing. In Proc. DATE, 2016.
[22] S. Sivanantham et al., "Low Power Floating Point Computation Sharing Multiplier for Signal Processing Applications," IJET 2013.
[23] G. Srinivasan et. al Significance Driven Hybrid 8T-6T SRAM for Energy-Efficient Synaptic Storage in Neural Networks, In Proc. DATE 2016.
[24] I. Chang et al. "A priority-based 6T/8T hybrid SRAM architecture for \ aggressive voltage scaling in video applications." Circuits and Systems for Video Technology, IEEE Transactions, 2011.
[25] S. Venkataramani et al. "AxNN: Energy-efficient neuromorphic systems using approximate computing." In Proc. ISLPED, 2014.
[26] J. Schmidhuber. Deep learning in neural networks. CoRR, 2014.
[27] S. Venkataramani et al. "Scalable-effort classifiers for energy-efficient machine learning." In Proc. DAC 2015.
[28] M. D Zeiler et. al. "Deconvolutional networks. In CVPR, 2010.
[29] J. Yosinski et. al. "How transferable are features in deep neural networks?" In NIPS, 2014.
[30] P. Panda et. al, "Conditional Deep Learning for Energy-Efficient and Enhanced Pattern Recognition", arXiv preprint arXiv:1509.08971.
[31] A. Sengupta et. al, "Spin-Transfer Torque Magnetic Neuron for Low Power Neuromorphic Computing", In Proc. IJCNN 2015.
[32] M. Sharad et. al, "Spin-Based Neuron Model with Domain Wall Magnets as Synapse," TNANO, 2012.
[33] M. Sharad et. al, "Spin Based Neuron-Synapse Unit for Ultra Low Power Programmable Computational Networks", In Proc. IJCNN 2012.
[34] A. Sengupta et. al, "Spin Orbit Torque Based Electronic Neuron", Applied Physics Letters, 2015
[35] A. Sengupta et. al, "Proposal for an All-Spin Artificial Neural Network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets", IEEE TBioCAS (In Press), arXiv:1510.00459.
[36] A. Sengupta et. al, "Short-Term Plasticity and Long-Term Potentiation in Magnetic Tunnel Junctions: Towards Volatile Synapses", Physical Review Applied 2016.
[37] C. F. Pai et. al, "Spin transfer torque devices utilizing the giant spin Hall effect of tungsten", Applied Physics Letters, 2012.
[38] I. M. Miron et. al, "Perpendicular switching of a single ferromagnetic layer induced by in-plane current injection", Nature 476, 2011.
[39] S. Emori, et. al, "Current driven dynamics of chiral ferromagnetic domain walls," Nature materials, 2013.
[40] R. Jolivet, et. al "Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy." Journal of neurophysiology, 2004.
[41] S. Mukhopadhyay et al. "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS." ,TCAD 2005.
[42] http://ptm.asu.edu
[43] A.Sengupta et. al "Spin-orbit torque induces spike-timing-dependent plasticity", Applied Physics Letters, 2015.
[44] D. Fan et. al "STT-SNN: A spin-transfer-torque based soft-limiting non-linear neuron for low-power artificial neural networks", TNANO 2015.
[45] T. Mytkowicz, "Programming Uncertain Things", In Proc. DAC 2016.
[46] M. Shafique, et. al, "Cross-Layer Approximate Computing: From Logic to Architectures", In Proc. DAC 2016.
[47] D. M. Mathew, et. al "Approximate Computing with Partially Unreliable Dynamic Random Access Memory: Approximate DRAM", In Proc. DAC2016.
[48] J. Henkel, "Approx. Computing: Solving Computing's Inefficiency Problem?" IEEE Design and Test, 2016.