Rectified-linear and Recurrent Neural Networks Built with Spin Devices

Qing Dong, Kaiyuan Yang, Laura Fick, David Blaauw, Dennis Sylvester
University of Michigan, Ann Arbor, MI
{qingdong, kaiyuan, lfreyman, blaauw, dmcs}@umich.edu

***Abstract*—A spin synapse with analog programmability using all charge current is proposed. Compared with using spin current, the proposed all-charge-current synapses can be placed in a larger cross-bar array to form a denser and larger neural network. Using the current summation, DOT product can be realized. We further employ a compact racetrack converter as the neuron to implement a rectified-linear neural network, saving area by 67% and energy by 69% compared with a spin binary-threshold neural network while achieving similar accuracy with MNIST digit recognition benchmark. Storing the domain wall motion in a time-based fashion, a recurrent neural network can be realized for time-involved inference tasks.**

Keywords—Domain Wall, Neural network, Spin, Recurrent

I. Introduction

CMOS implementations of neural network suffer from large area for weight storage and high standby power. Spin devices such as magnetic tunnel junction (MTJ), domain wall (DW) and racetrack nanowires have demonstrated great potential for memory [1-3], logic [4], and analog computation [5] due to their non-volatility, zero standby power and compact area, making them promising candidates as neural network components. There have been several recent works on spin neural networks. Lateral-spin-valve neuron [6] only uses binary weights and suffers from the short diffusion distance of spin current. The DW binary-threshold neural network [7], combining with RRAM or PcRAM, requires more masks and complicates fabrication. DW synapse [8] has analog programmability but its spin current limits the number of synapses connected with neurons. None of them realized analog programmability with all charge current. Also, they are all feed-forward binary-threshold neural networks (BTNN), which are limited in their effectiveness in that more neurons and hidden layers are required to perform the same function compared with a rectified-linear neural network (RLNN). Time-related neural networks such as recurrent neural network (RNN) have not been explored yet.

We propose a spin synapse device with analog programmability using all charge current. In this device, the resistance can be varied in an analog fashion according to the DW position, which can be moved by charge current injection instead of spin current diffusion. The proposed synapse devices are placed in a cross-bar array configuration to form a dense neural network. Using current summation on the bit-line, DOT product function can be realized. Using an ultra-compact racetrack converter [5], we can build a more efficient RLNN. A novel majority voting circuit is proposed in the final-decision layer for recognition tasks. The spin RLNN reduces area by 67% and energy by 69% compared to spin BTNN with equal error rates. Furthermore, integrating with time, current-induced DW motion can be used for time-related analog storage to form recurrent neuron.

II. Components of Spin Neural Network

A. Spin Synapse

To realize analog programmability with all charge current, we propose a spin synapse device as shown in Fig. 1(a). Due to current-induced DW motion, the DW can be moved by charge current flowing through horizontal ports T_{PROG0} and T_{PROG1}. Moreover, the DW moving distance X is linearly proportional to the current I or the time T [5,10], expressed as:

$$\boldsymbol{X = v * T = \frac{\beta\mu P}{\alpha e M_s}\left(\frac{I}{Area} - J_{th}\right) * T} \tag{1}$$

The vertical conductance between the two ports T_{READ0} and T_{READ1} can be treated as the parallel connection of spin parallel conductance $G_P(1\text{-}X)$ and spin antiparallel conductance $G_{AP}X$ (Fig. 1(b)), both of which are determined by the DW position X. Therefore, DW motion can change the vertical conductance from G_P to G_{AP} (Fig. 1(b)) in an analog fashion [11]. We develop a Verilog-A model of the proposed synapse device that describes the relationship between vertical conductance and DW position as shown in Fig. 1(c). Previous spin current synapses [6-8] suffer from short spin diffusion distance in the spin channel, and thus large scale connections between synapses and neurons are impossible. As no spin current is involved in the program and sensing operations of our proposed synapse, larger scale interconnections between synapses and neurons can be realized.

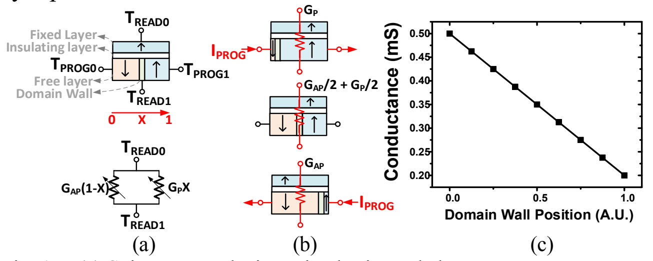

Fig. 1. (a) Spin synapse device using horizontal charge current to program the DW position in an analog way. (b) The vertical conductance of spin synapse device changes from G_P to G_{AP} according to the DW position. (c) Verilog-A model of the domain wall position *vs.* conductance.

B. Racetrack Converter as Rectified-linear Neuron

BTNN can use only a simple comparator neuron while an RLNN requires an analog-to-digital converter (ADC) as the neuron. The large area of CMOS ADCs limits their use in RLNN. A spin-based ADC [5] was proposed with 1000× smaller area than state-of-art CMOS ADCs. Due to its ultra-compact area and high energy efficiency, the racetrack converter is an ideal neuron for RLNN. Fig. 2 shows a 3-bit ADC using three racetrack nanowires. Each nanowire can be configured differently with a sequence of alternating write and shift current pulses, such that each generates a single bit, from LSB to MSB. During conversion, the input current can move all the DWs simultaneously. After a fixed time, the DW moving distance is determined by the input current value. Then, by sensing the resistance of the read MTJ head above each nanowire, the data can be read out as a digital value. In this way, the analog current is converted into a digital value through this racetrack converter. After read, the racetrack converter can be reset for next cycle as described in [5].

C. Recurrent DW Neuron

RNN requires analog storage and the analog value must be able to accumulate cycle by cycle. CMOS implementations typically use a charge-storing capacitor as a recurrent neuron, however this suffers from large area and leakage. With current-induced DW motion, DW can be a desirable analog storage element, memorizing and accumulating analog value. Fig. 4 (bottom right) shows a simple recurrent DW neuron device. The DW starts at position $Y_{(t)}$. In the first cycle, the DW is moved by $\Delta Y_{(t+1)}$, and stops at $Y_{(t+1)}$. The device stores this DW position, and the output remains at 0 because the spin polarity of top and bottom MTJ layers are the same. But in the second cycle, the DW is moved to $Y_{(t+2)}$ from $Y_{(t+1)}$ and the output changes to 1 due to polarity flip of the bottom free layer. Therefore, the output of the neuron device is determined by not only the current input but also previous state. And thus the recurrent DW neuron can handle time-related inference tasks. Our example shows a simple recurrent DW neuron device with only binary-threshold output. The racetrack converter can be modified to create a recurrent neuron with linear output functionality, because it also uses current-induced DW motion for conversion.

III. Neural Network Architecture

A. Current Summation for DOT Product

As shown in Fig. 3(a), neural network function can be divided into DOT product and processing. Binary-threshold is the most common neuron function in hardware because of its simplicity. However, BTNN requires many more synapses and neurons to perform similar task compared to other complicated neural networks. RLNN employs an ADC as the neuron instead of a single comparator, improving its capabilities. Both BTNN and RLNN are feed-forward neural networks which can only work on static tasks like digit recognition. To deal with time-related tasks like forecasting and phoneme recognition, conventional time-related neural networks use a shift-register at the inputs and shift the inputs cycle by cycle to add time information into the system, which are inefficient in terms of area, latency and power. Recurrent neuron itself can store the time information as internal state and generate the results based on both current inputs and previous state. Therefore, RNN can be applied to highly-efficient time-related inference.

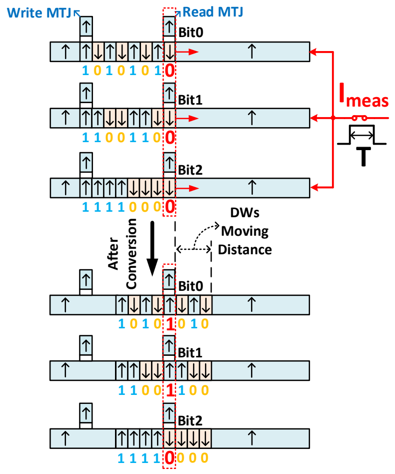

Fig. 2. Structure of a 3b racetrack converter. Each nanowire is configured with different DWs granularity, representing an individual bit. During conversion, DWs move simultaneously and stop at a distance that is proportional to input current.

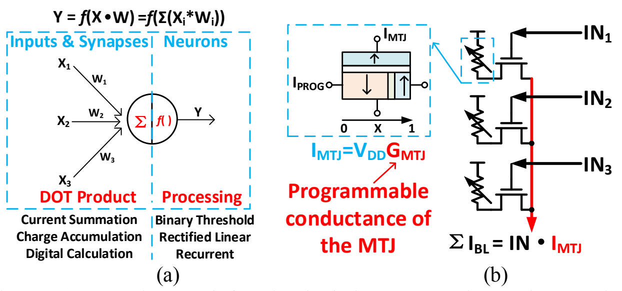

Fig. 3. (a) Neural network function includes DOT product and processing. (b) Current summation on bit-line for DOT product of inputs and weights.

DOT product can be implemented by current summation, charge accumulation or digital calculation (Fig. 3(a)). Charge accumulation suffers from leakage and digital calculation requires complicated ALU and large intermediate storage. Current summation is more accurate and suitable for analog computing. As shown in Fig. 3(b), each spin synapse is connected to a NMOS. Input signal can turn on/off the NMOS. If the NMOS is on, the synapse will generate some current onto the bit-line, and all currents will be summed together on the bit-line. The current value is determined by the DW position, representing the weight. The summed current value is the DOT product of *IN* and I_{MTJ}. As a static operation, current summation is immune to dynamic noise.

B. Cross-bar Array Neural Network Configuration

The proposed spin synapse works with all charge current, and thus multiple synapses can be connected to one bit-line, enabling massive cross-bar array configurations. Fig. 4 shows the cross-bar synapse array structure. One spin synapse and one NMOS access device make up one cell in the array. Inputs are the word-lines of the array; weights are represented by the programmable conductance of the spin synapse device. Before any neural network operation, the weights should be programmed by horizontal current injection with varying finely controlled pulse width.

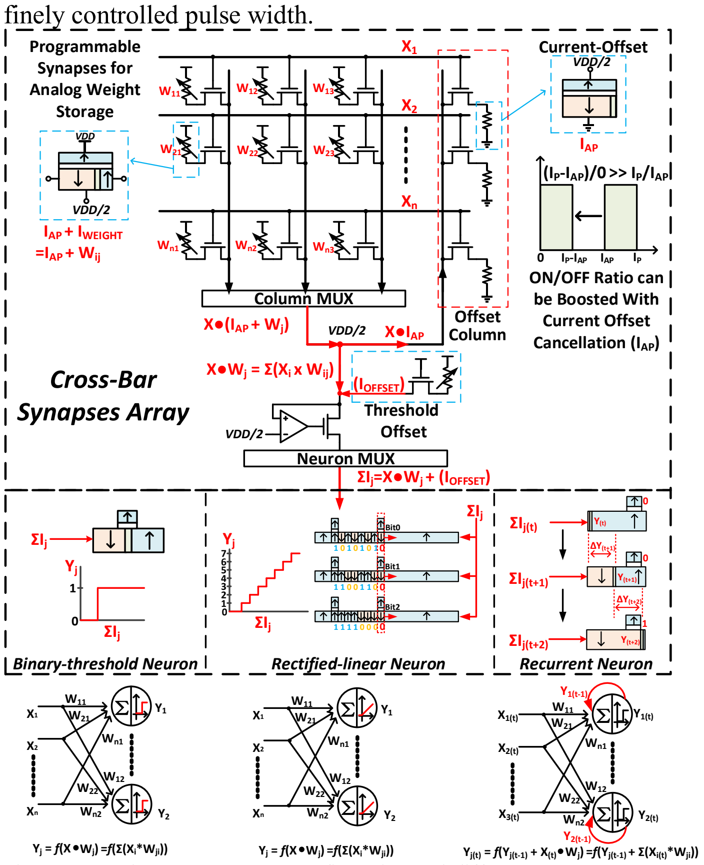

Fig. 4. Cross-bar synapse array configuration and different neuron types. Offset column is used to improve on/off ratio. Mathematical models for BTNN, RLNN and RNN are shown in the bottom.

Current summation on each bit-line performs a DOT product of inputs and weights to calculate each neuron Y_j. Neurons in a single layer are calculated in a serial fashion by adding a column MUX and a neuron DEMUX and iterating their addresses. Current of a single synapse varies in the range from I_{AP} to I_P, which is barely a 2.5× range. To amplify the on/off ratio of the unit synapse current, we add one extra offset column to provide offset current I_{AP}, which increases the on/off ratio from I_P/I_{AP} to $(I_P-I_{AP})/0$. With column MUX, only one offset column is required for a full array. We also add an analog buffer to fix the node voltage at VDD/2, such that the I_{AP} is equal on both summation and offset bit-lines. The residue current represents the DOT product results and flows into a neuron selected by neuron DEMUX for processing. The mathematic models for BTNN, RLNN and RNN are shown in the bottom of Fig. 4.

A DW binary threshold neuron works as a current comparator. Once the residue current exceeds the threshold, the DW will move and flip the spin polarity of the bottom free layer. The proposed rectified linear neuron then uses a racetrack ADC to convert the analog residue current into a digital value and this converted digital value serves as the inputs for next layer, fully utilizing the analog residue current. For the recurrent neuron, Fig. 4 also shows a simple recurrent neuron, storing time-related analog information as DW location. In each cycle, DW will move and stop, and the distance is proportional to the residue current. In each cycle, the neuron generates a digital output. When the DW moves to the right, the neuron will be activated and output a 1.

As the current-induced DW motion has a threshold, the binary-threshold neuron can use it as the reference. But both rectified-linear neuron and recurrent neuron require some extra offset for this threshold. This can be realized by adding an extra threshold offset device on the current subtraction node to generate a required threshold current I_{OFFSET} onto the residue current (Fig. 4). And the DW position inside this threshold offset device should be programmed according to the current threshold of the neuron once after fabrication.

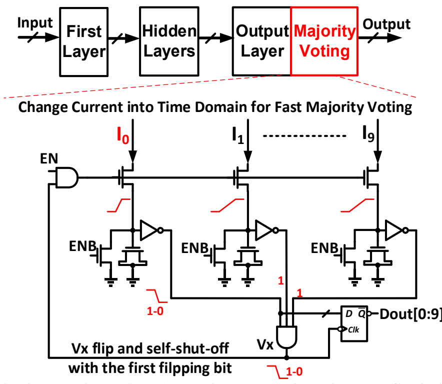
Input
First Layer
Hidden Layers
Output Layer
Majority Voting
Output
Change Current into Time Domain for Fast Majority Voting
I0
I1
I9
EN
ENB
ENB
ENB
1-0
1
1
Dout[0:9]
Vx
Vx flip and self-shut-off with the first filpping bit
1-0

Fig. 5. Majority voting circuit. Using cap charging to find the maximum value and make final decision for recognition task.

C. *Majority Voting Circuit*

A typical neural network system includes an input layer, several hidden layers, and an output layer (Fig. 5). For classifier applications, the output layer uses a majority voting circuit to find the maximum value for final decision. Conventional comparison methods need many comparators and cycles to find such a maximum value. Some winner-take-all circuits employ complicated analog modules. Here, we propose a simplified majority voting circuit working in time domain that more easily finds a peak value compared with using current or voltage domain. As shown in Fig 5, all dedicated NMOS capacitors are discharged to ground initially. Then after signal EN goes high, the highest current path will flip the inverter first and shut off all the switches. The 0 output of the inverter represents the highest current. The operation only takes one cycle. Using just small NMOS gate capacitors the design is area-efficient and low-power.

D. *Extending the RLNN layer to multi-bit input*

In RLNN, input of the first layer is 1-bit; while the following layers take 3-bit inputs. For 3-bit input DOT product, each bit of the input will be calculated separately as in 1-bit input case. Each bit of the input fires one WL and 3-bit input takes 3 WLs (Fig. 6). Also, 3 BLs are used for one neuron. 3 by 3 cells represent one weight: 3 of them on the diagonal have same conductance as weight; others can be programmed to provide offset. During operation, each BL will sum up the currents, then current mirrors with 1X, 2X and 4X ratios will be used to get the real total current for neurons in hidden layers or majority-voting circuit in the output layer.

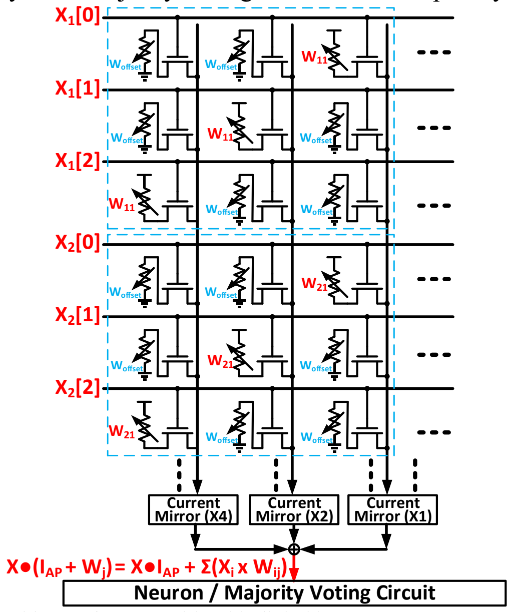

Fig. 6. Final layer of RLNN with 3-bit digital inputs.

I. Simulation Results and Analysis

We develop compact Verilog-A models for related MTJ, DW, and racetrack nanowire based on published experimental data [3, 10, 12-13]. Co-simulation with CMOS circuits (32nm technology) is performed in SPICE. We built both spin-based BTNN and RLNN containing one input layer, one hidden layer and one output layer. Fig. 7 shows the parameters of the synapse and neuron [5, 8, 10]. We use complete 60000 MNIST digit recognition training sets to train both BTNN and RLNN, and use 10000 standard MNIST test sets to evaluate the error rates. Fig. 8 (a) and (b) show the error rate decreasing with training epochs for both BTNN and RLNN with the same number of hidden neurons (100). For BTNN, 8-bit quantized weight has little difference with floating-point weight because the binary-threshold function itself generates substantial quantization error and thus adding weight bits does not reduce error rate. However in RLNN additional weight bits serve to improve error rate. With 100 hidden neurons, the RLNN can achieve 2.5× error rate reduction compared to BTNN (Fig. 8(c)). Fig. 8(d) compares the error rate change with synapse weight bits for both BTNN and RLNN. RLNN shows better error rate scaling with number of weight bits. Fig. 8(e) shows the error rate decreasing with more hidden neurons. To achieve <5% error rate, the RLNN needs only 30 hidden neurons; while BTNN requires 100 hidden neurons. As RLNN uses an ADC as the neuron instead of a single comparator, the error rate can be reduced with more ADC resolution (Fig. 8(f)). In our experiments, a 3-bit ADC is sufficient to achieve good error rates for RLNN. Table I compares CMOS SRAM-based BTNN, spin BTNN, and spin RLNN with the same synapse weight (4 bit) and same error rate. Area and energy of peripheral circuits are included. While achieving the same accuracy of MNIST task, spin-based BTNN saves area and energy by 96% and 14% respectively, compared with CMOS SRAM-based BTNN. The proposed spin RLNN further reduces area by 67% and energy by 69% compared with spin BTNN.

We take a simple phoneme recognition task as a training example to show the effectiveness of spin RNN. As shown in Fig. 9(a), the input is a mixed phoneme waveform in which one specific frequency needs to be recognized, and the output of the neural network remains high once detecting this specific frequency. Conventional feed-forward BTNN can deal with this task with time-lagged inputs by adding shift registers. Fewer hidden neurons are required as more input delay stages of the shift registers are added (Fig. 9(b)). However, to solve the same task, RNN requires only 3 hidden neurons and 5 cycles, significantly more effective than conventional time-lagged feed-forward neural network.

Item	Value
Technology	32nm
R_{AP}	5KΩ
R_P	2KΩ
Vdd	100mV
I_{AP}	10uA
I_P	25uA
Frequency	10MHz
Synapse Size	0.027 um^2 (30F^2)
Binary-threshold Neuron Size	0.4 um^2
Rectified-linear Neuron Size	2 um^2
Majority Voting Circuits Area	50 um^2
Synapse Weight Bits	3
Current Resolution	0.02uA

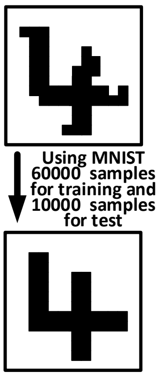

Fig. 7. Device parameters used for co-simulation with CMOS; Complete MNIST digit recognition benchmark are used for training evaluation.

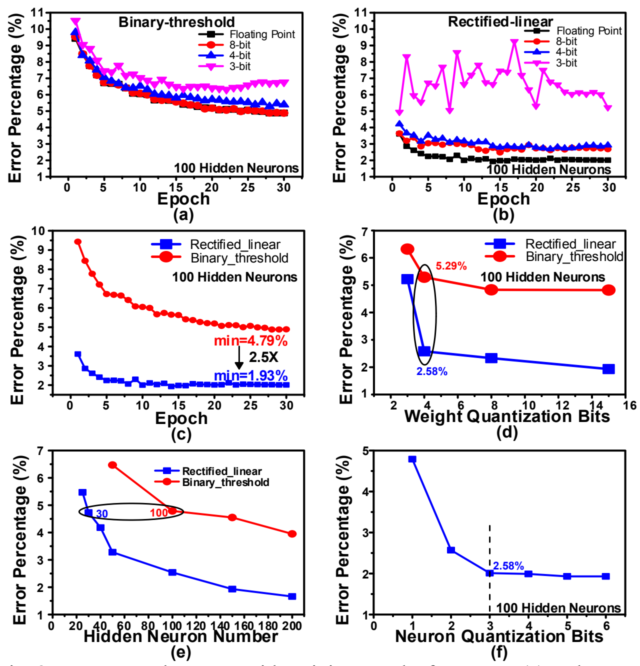

Fig. 8. Error rate decreases with training epochs for BTNN (a) and RLNN (b). RLNN achieves 2.5× error rate reduction compared with BTNN (c). Error rate can be reduced with more weight bits (d) and hidden neurons (e). Higher resolution of ADC neuron can further lower error rate of RLNN (f).

For memory applications, multi-bit MRAM is challenging due to limited sensing margin and cell variation. But in neural network application, each BL has hundreds of cells and each neuron only need to distinguish 1-bit or 3-bit output of summed current. Take 100 3-bit synapses on one BL as example, each cell generates 1-8 unit currents. For memory application, sense amplifier has to distinguish each unit of current. While in neural network, the 3-bit rectified-linear neuron only need to distinguish 100/200/…800 rather than 1/2/…8. Therefore, the sensing margin is not a problem for neural network application. Current summation of large number of cells can average out the random variation of cell current. For systematic variation, the actual measured results will show some bias, according to which the offset added to each column can be adjusted to compensate the systematic variation. Moreover, DNN algorithm is inherently variation-tolerant. Once systematic offset of different weights is measured, it could be incorporated in the training to compensate for it by adjusting the weights.

Table I. Comparison among CMOS BTNN, Spin BTNN and Spin RLNN.

	CMOS BTNN	Spin BTNN	Spin RLNN
Synapse Weight	4	4	4
Hidden Neurons	100	100	30
Error Rate	4.79 %	4.79 %	4.73 %
Area	89000 um^2	3700 um^2	1220 um^2
Energy per Frame	16.6 nJ	14.2 nJ	4.4 nJ
Energy per pixel	21 pJ	18 pJ	5.6 pJ

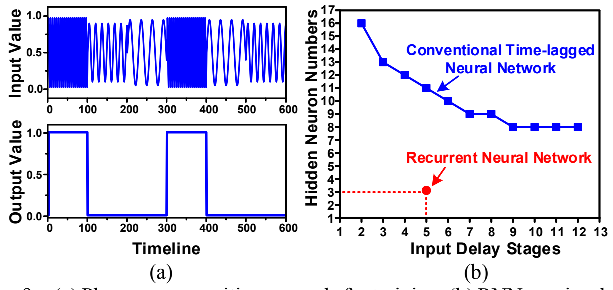

Fig. 9. (a) Phoneme recognition example for training; (b) RNN requires less hidden neurons and shorter latency than time-lagged neural network.

II. Conclusions

A spin synapse device has been proposed with analog programmability using all charge current. The synapse devices can be placed in a cross-bar array to form a dense neural network. DOT product can be realized using current summation. With compact racetrack converter as the neuron, spin RLNN is implemented, which saves area by 67% and energy by 69% compared to spin BTNN with equal error rate. Storing the DW motion in a time-based fashion, the RNN can also be realized for time-involved inference tasks. Compared to conventional time-lagged feed-forward neural network, RNN requires fewer hidden neurons and latency cycles.

References

[1] K. C. Chun, et al., "A Scaling Roadmap and Performance Evaluation of In-Plane and Perpendicular MTJ Based STT-MRAMs for High-Density Cache Memory," IEEE JSSC, pp. 598-610, 2013.

[2] S. Fukami, et al., "High-speed and reliable domain wall motion device: Material design for embedded memory and logic application," VLSIT, 2012.

[3] L. Thomas, et al., "Racetrack Memory: a high-performance, low-cost, non-volatile memory based on magnetic domain walls," IEDM, 2011.

[4] D. Morris, et al., "mLogic: Ultra-Low Voltage Non-Volatile Logic Circuits Using STT-MTJ Devices," DAC, 2012.

[5] Q. Dong, et al., "Racetrack Converter: Low Power and Compact ADC/TDC Using Racetrack Spintronic Devices," IEEE ISCAS, 2015.

[6] M. Sharad, et al., "Spin Neuron for Ultra Low Power Computational Hardware," Device Research Conference, 2012.

[7] M. Sharad, et al., "Ultra Low Power Associative Computing with Spin Neurons and Resistive Crossbar Memory," DAC, 2013.

[8] M. Sharad, et al., "Spin-Based Neuron Model with Domain-Wall Magnets as Synapse," Trans. on Nanotechnology, pp. 843-853, 2012.

[9] V. Calayir, et al., "All-Magnetic Analog Associative Memory," NEWCAS, 2013.

[10] Y. Zhang, et al., "Perpendicular-magnetic-anisotropy CoFeB racetrack memory," Journal of Applied Physics, 2012.

[11] X. Wang, et al., "Spintronic Memristor through Spin-Torque-Induced Magnetization Motion," EDL, 2009.

[12] D. Chiba, et al., "Control of Multiple Magnetic Domain Walls by Current in a Co/Ni Nano-Wire," Applied Physics Express, 2010.

[13] S. Fukami, et al., "Low-Current Perpendicular Domain Wall Motion Cell for Scalable High-Speed MRAM," VLSIT, 2009.