An Energy-Efficient Nonvolatile In-Memory Computing Architecture for Extreme Learning Machine by Domain-Wall Nanowire Devices

Yuhao Wang, Hao Yu, *Senior Member, IEEE*, Leibin Ni, Guang-Bin Huang, *Senior Member, IEEE*, Mei Yan, Chuliang Weng, Wei Yang, and Junfeng Zhao

***Abstract*—The data-oriented applications have introduced increased demands on memory capacity and bandwidth, which raises the need to rethink the architecture of the current computing platforms. The logic-in-memory architecture is highly promising as future logic-memory integration paradigm for high throughput data-driven applications. From memory technology aspect, as one recently introduced nonvolatile memory device, domain-wall nanowire (or race-track) not only shows potential as future power efficient memory, but also computing capacity by its unique physics of spintronics. This paper explores a novel distributed in-memory computing architecture where most logic functions are executed within the memory, which significantly alleviates the bandwidth congestion issue and improves the energy efficiency. The proposed distributed in-memory computing architecture is purely built by domain-wall nanowire, i.e., both memory and logic are implemented by domain-wall nanowire devices. As a case study, neural network-based image resolution enhancement algorithm, called DW-NN, is examined within the proposed architecture. We show that all operations involved in machine learning on neural network can be mapped to a logic-in-memory architecture by nonvolatile domain-wall nanowire. Domain-wall nanowire-based logic is customized for in machine learning within image data storage. As such, both neural network training and processing can be performed locally within the memory. The experimental results show that the domain-wall memory can reduce 92% leakage power and 16% dynamic power compared to main memory implemented by DRAM; and domain-wall logic can reduce 31% both dynamic and 65% leakage power under the similar performance compared to CMOS transistor-based logic. And system throughput in DW-NN is improved by 11.6x and the energy efficiency is improved by 56x when compared to conventional image processing system.**

***Index Terms*—Domain wall, extreme learning machine, in-memory computing, nonvolatile memory.**

Manuscript received July 20, 2014; revised March 27, 2015; accepted June 12, 2015. Date of publication June 19, 2015; date of current version November 6, 2015. This work was supported by Singapore NRF-CRP (NRF-CRP9-2011-01), MOE Tier-2 (MOE2010-T2-2-037 (ARC 5/11)), A*STAR PSF fund 11201202015 and Huawei Shannon Research Lab. The review of this paper was arranged by Associate Editor C. A. Moritz.

Y. Wang, H. Yu, L. Ni, G.-B. Huang, and M. Yan are with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore (e-mail: ywang29@e.ntu.edu.sg; haoyu@ntu.edu.sg; NILE0001@e.ntu.edu.sg; EGBHUANG@NTU.EDU.SG; YanMei@ntu.edu.sg).

C. Weng, W. Yang and J. Zhao are with Shannon Laboratory, Huawei Technologies Co., Ltd., Hangzhou, 310051 China (e-mail: wengchuliang@huawei.com; william.yangwei@huawei.com; junfeng.zhao@huawei.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNANO.2015.2447531

I. Introduction

THE analysis of big-data at exascale (10^{18} bytes/s or flops) has introduced the emerging need to reexamine the existing hardware platform that can support memory-oriented computing. A big-data-driven application requires huge bandwidth with maintained low-power density. The most widely existed data-driven application is machine learning in big data storage system, as the most exciting feature of future big-data storage system is to find implicit pattern of data and excavate valued behavior behind. Take image searching as an example, instead of performing the image search by calculating pixel similarity, image search by machine learning is a similar process as human brains, which learns the features of all images by feature extraction algorithms and compares the features in the form of strings. As such, the image search becomes a traditional string matching problem which is much easier to solve. However, to handle big image data at exa-scale, there is a memory wall that has long memory access latency as well as limited memory bandwidth. Again take the example of image search in one big-data storage system, there may be billions of images, so that to perform feature extraction for all images will lead to significant congestion at I/Os when migrating data between memory and processor. In addition, the large volume of memory will experience significant leakage power, especially at advanced CMOS technology nodes, for holding data in volatile memory for fast accesses [1], [2].

From memory technology point of view, there are many recent explorations by the emerging non-volatile memory (NVM) technologies at nano-scale such as phase-change memory, spin-transfer torque memory (STT-RAM), and resistive memory (ReRAM) [3]–[7]. The primary advantage of NVM is the potential as the universal memory with significantly reduced leakage power. For example, STT-RAM is considered as the second-generation of spin-based memory, which has sub-nanosecond magnetization switching time and sub-pJ switching energy [8]–[10]. As the third-generation of spin-based memory, domain-wall nanowire, also known as racetrack memory [11], [12], is a newly introduced NVM device that can have multiple bits densely packed in one single nanowire, where each bit can be accessed by the manipulation of the domain-wall. Compared with STT-RAM, the domain-wall nanowire is able to provide the similar speed and power but with much higher density or throughput [13]. Since domain-wall nanowire has close-to-DRAM density but with close-to-zero standby power, it becomes an ideal

candidate for future main memory that can be utilized for big-data processing.

From architecture point of view, the logic-in-memory architecture is introduced to overcome memory bandwidth issue [14]–[18]. The basic idea behind is that, instead of feeding processor large volume of raw data, it is beneficial to preprocess the data and provide processor only intermediate result. In other words, the key is to lower communication traffic by operands reduction. For example, to perform a sum of ten numbers, instead of transmitting ten numbers to processor, in-memory architecture is able to obtain the sum by in-memory logic and transmit only one result thus reduce traffic by 90%. To perform in-memory logic, it is necessary to implement logic inside memory so that preprocessing logic can be done. However, the in-memory logic circuits in current approaches are composed of CMOS transistors, which are usually made simple otherwise the power and area overhead would be overwhelming. Interestingly, domain-wall nanowire device not only has the potential for high density and high performance memory design, but also interesting computing capability due to spin-physics. Therefore, it is very promising to implement an in-memory architecture with both non-volatile domain-wall memory and non-volatile in-memory logic. Such memory based in-memory logic may overcome the functionality limit of transistor based ones.

However, currently there is no in-depth study to explore domain-wall nanowire based in-memory computing architecture. For example, no link has been made to perform big-data logic operation based on spin-based device such as domain-wall nanowire. What is more, no domain-wall nanowire device model has been developed in terms of accuracy and efficiency for circuit designs. In this paper, the image processing algorithm by neural network learning is examined within the domain-wall nanowire based in-memory architecture. The contributions of this work are: firstly, a SPICE behavioral model of domain-wall nanowire has been developed for circuit-level verification of both memory and logic designs; secondly, the domain-wall memory has been explored as low power main memory; thirdly, we show that physics of spintronics of domain-wall nanowire can introduce unique capability to perform logic operations such as XOR and addition which other NVM devices do not have; lastly, a purely domain-wall memory based distributed in-memory computing architecture is proposed, and we show the feasibility of mapping the ELM neural network to the proposed architecture, called DW-NN. The numerical experiments show that, the domain-wall memory can reduce 92% leakage power and 16% dynamic power compared to main memory implemented by DRAM; and domain-wall logic can reduce 31% both dynamic and 65% leakage power under the similar performance compared to CMOS transistor based logic. And compared to the scenario that ELM is executed in CMOS based general purpose processor, the proposed DW-NN improves the system throughput by 11.6x and energy efficiency by 56x.

The rest of this paper is organized in the following manner. Section II describes the overall in-memory computing platform based on domain-wall nanowire. Section III discusses the physics and SPICE model of domain-wall nanowire. Section IV details the main memory design by domain-wall nanowire. Section V presents the domain-wall XOR, addition, and general LUT by domain-wall nanowire. Section VI introduces the mapping of machine learning based super-resolution algorithm on the proposed distributed in-memory architecture. Experimental results are presented in Section VII with conclusion in Section VIII.

II. Nonvolatile In-Memory Computing Platform

Conventionally, all the data is maintained within memory that is separated from the processor but connected with I/Os. Therefore, during the execution, all data needs to be migrated to processor and written back afterwards. In the data-oriented applications, however, this will incur significant I/O congestions and hence greatly degrade the overall performance. In addition, significant standby power will be consumed in order to hold the large volume of data.

Theoretically, it is feasible to overcome the bandwidth issue by adding more I/O pins or operating them at higher frequency. Practically, however, the I/O frequency is limited by the signal propagation delay and signal integrity issues, and I/O number is limited by the packaging technology, thus the bandwidth can hardly get further improved. Instead of improving memory bandwidth, it is also possible to reduce the required data communication traffic between memory and processor. The basic idea behind is that, instead of feeding processor large volume of raw data, it is beneficial to preprocess the data and provide processor only intermediate result. The key to lower communication traffic is the operands reduction. For example, to perform a sum of ten numbers, instead of transmitting ten numbers to processor, in-memory architecture is able to obtain the sum by in-memory logic and transmit only one result thus reduce traffic by 90%. To perform in-memory logic, it is necessary to implement logic inside memory so that preprocessing logic can be done. Such architecture is called logic-in-memory architecture. Considering the leakage reduction at the same time, logic-in-memory architectures that are associated with NVM are presented in [14]–[17], [19]. Fig. 1 shows logic in memory architecture at memory cell level. The example illustrated here is an in-memory full adder with both *sum* logic and *carry* logic.

The basic circuitry, including access transistor, the word-line and bit-lines, is to ensure memory access. The data is stored in NVM devices which have either low or high resistance. Redundant data is required for each bit of data for logic purpose. Combinational logic circuit is added inside which the non-volatile devices are equivalent to transistors: considered turned on if at low resistance state or turned off if at high resistance state. In such architecture, the desired result can be obtained immediately without reading operands as if the results are already stored in data array and it is just be 'readout'. This is very useful for some specific applications as this architecture is able to preprocess data without loading data to processor with extremely short latency.

As the logic is inserted to one cell or a few cells, it is limited to small size thus cannot be made complex. Usually only simple logic is suitable for such architecture otherwise the overhead would be overwhelming. Though simple logic in such

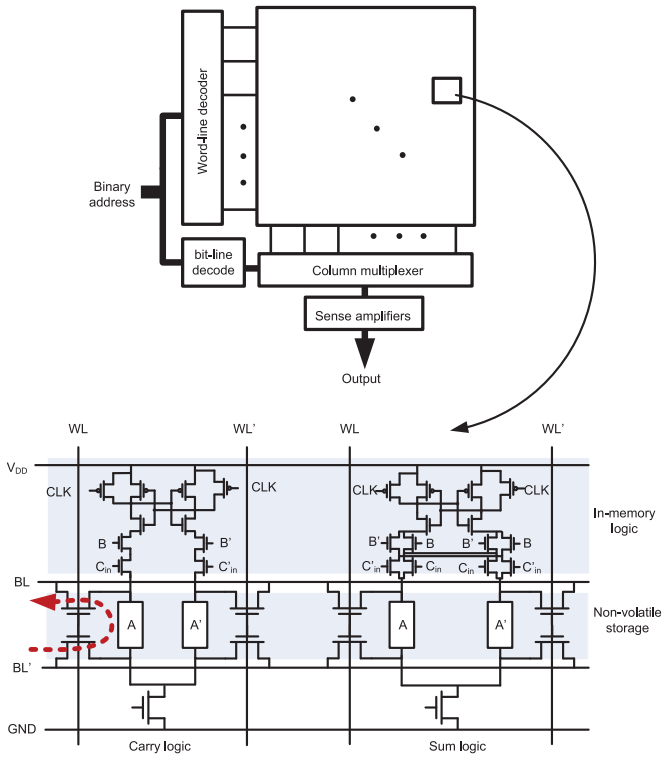


Fig. 1. In-memory computing architecture at memory cell level.

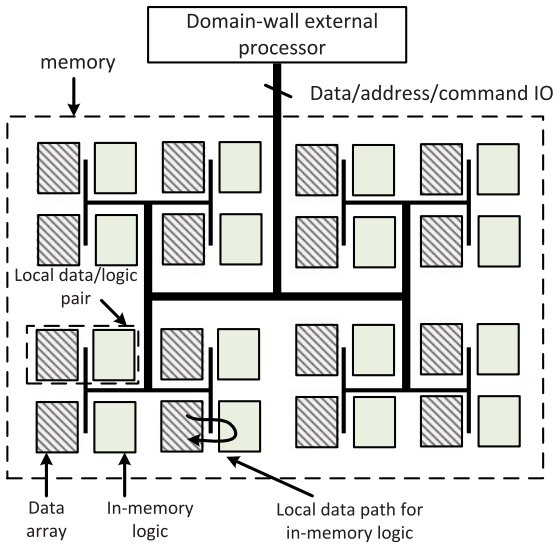


Fig. 2. In-memory computing architecture at memory block level.

architecture is able to share the workload of processor, its effect to reduce communication traffic is not obvious due to limited operands reduction. In addition, similar to the operation of memory, for the whole data array only a few logic can be active concurrently at one time. This leads many logic circuits to be idle at most of the time, which is not only a waste of computational resources but also incurs leakage power for CMOS logic. Another disadvantage is that the data needs to be stored in a very strict manner, determined by in-memory logic circuit functionality.

An alternative in-memory architecture we are presenting at block level in distributed fashion is illustrated in Fig. 2, which is more effective for traffic reduction. A memory data is usually organized in H-tree fashion, and the data block can be the data array or a number of data arrays that belong to same 'H-tree' branch. Instead of inserting in-memory logic at memory cell level inside the data array, the architecture in Fig. 2 pairs each block of data with in-memory logic (accelerators). Different from the cell level in-memory architecture, the accelerators can be made with higher complexity, and the number of accelerators for each data block can also be customized. The data flow of the block level in-memory architecture is to readout data from data block to in-memory logic, which performs particular functionality and then writes back the result. The data also needs to be stored in assigned blocks but it is much more flexible than that of cell level in-memory architecture. The block level in-memory architecture is very effective to reduce communication traffic between memory and processor. This is because significant operands reduction can be achieved by deploying accelerator with high level functionality. For example, for face recognition in image processing application, instead of transmitting a whole image to obtain a Boolean result, the result can be directly gained through in-memory logic. In other words, the block level in-memory architecture is suitable for big data driven applications where traffic reduction is more important than latency reduction.

In this platform, the domain-wall nanowire is intensively used. Both the memory block and logic block in each pair are purely implemented by domain-wall nanowire devices. In addition, energy efficient domain-wall logic units are deployed in the external processor to execute instructions that cannot be accelerated by in-memory logic. The domain-wall memory design will be discussed in details in Section IV, and domain-wall logic design will be presented in Section V.

III. Device Model and Simulation of Domain-Wall Nanowire

Domain-wall nanowire, also known as racetrack memory [11], [13], [20], is a newly introduced NVM device in which multiple bits of information are stored in single ferromagnetic nanowire. As shown in Fig. 3(a), each bit is denoted by the leftward or rightward magnetization direction, and adjacent bits are separated by domain walls. By applying a current through the shift port at the two ends of the nanowire, all the domain walls will move left or right at the same velocity while the domain width of each bit remains unchanged, thus the stored information is preserved. Such a tape-like operation will shift all the bits similarly like a shift register.

In order to access the information stored in the domains, a strongly magnetized ferromagnetic layer is placed at desired position of the ferromagnetic nanowire and is separated by an insulator layer. Such a sandwich-like structure forms a magnetic-tunnel-junction (MTJ), through which the stored information can be accessed. In the following, the write, read and shift operations are modeled respectively.

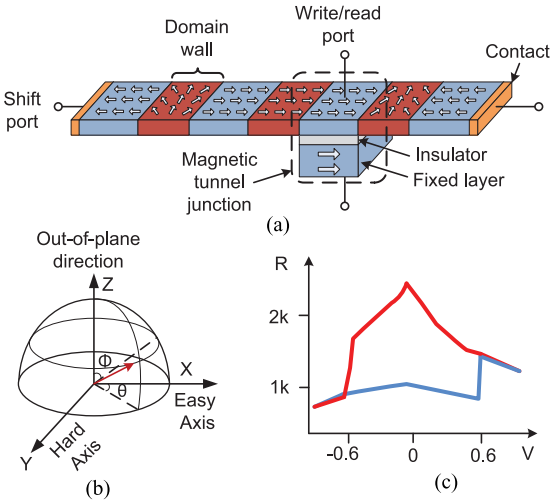


Fig. 3. (a) Schematic of domain-wall nanowire structure with access port and shift port; (b) magnetization of free-layer in spherical coordinates with defined magnetization angles; and (c) typical R-V curve for MTJ.

A. Magnetization Reversal

The write access can be modeled as the magnetization reversal of MTJ free layer, i.e. the target domain of the nanowire. Note that the dynamics of magnetization reversal can be described by the precession of normalized magnetization m, or state variables θ and ϕ in spherical coordinates as shown in Fig. 3(b). The spin-current induced magnetization dynamics described by θ and ϕ is given by [21]

$$\theta = \theta_0 \text{Exp}\left(-\frac{t}{t_0}\right) \cdot \cos(\phi) \tag{1}$$

$$\omega = \frac{d\phi}{dt} = k_1\sqrt{k_2 - (k_3 - k_4 I)^2} \tag{2}$$

where θ_0 is the initial value of θ, slightly tilted from the stable x or $-x$ directions; t_0 is procession time constant; ω is the angular speed of ϕ; k_1 to k_4 are magnetic parameters with detailed explanation in [21]; and I is the spin-current that causes the magnetization precession.

B. Magnetic-Tunnel-Junction Resistance

A typical R-V curve for MTJ is shown in Fig. 3(c) with two regions: giant magnetoresistance (GMR) region and tunneling region. Depending on the alignment of magnetization directions of the fixed layer and free layer, parallel or anti-parallel, the MTJ exhibits two resistance values R_l and R_h. As such, the general MTJ resistance can be calculated by the giant magnetoresistance (GMR) effect

$$R(\theta_u, \theta_b) = R_{l0} + \frac{R_{h0} - R_{l0}}{2}(1 - \cos(\theta_u - \theta_b)) \tag{3}$$

where θ_u and θ_b are the magnetization angles of upper free layer and bottom fixed layer, R_{l0} and R_{h0} are the MTJ resistances when the applied voltage is subtle. When the applied voltage increases, there exists tunneling effect caused voltage-dependent resistance roll-off,

$$\begin{cases} R_l(V) = \dfrac{R_{l0}}{1 + c_l V^2} \\ R_h(V) = \dfrac{R_{h0}}{1 + c_h V^2} \end{cases} \tag{4}$$

where c_l and c_h are voltage-dependent coefficients for parallel state and anti-parallel states, respectively.

C. Domain-Wall Propagation

Like a shift register, the domain-wall nanowire shifts in a digital manner, thus could be digitalized and modeled in the unit of domains, in which a bit is stored. The magnetization orientations in adjacent domains can be either parallel or anti-parallel, and the magnetization transition occurs in the connecting domain-wall in case of anti-parallel neighboring bits. Therefore, a domain plus a domain-wall are the basic unit for the shift-operation. Note that except the bit in the MTJ, the other bits denoted by the magnetization directions are only affected by their adjacent bits. In other words, the magnetization of each bit is controlled by the magnetization in adjacent domains. Inspired by this, we present a magnetization controlled magnetization (MCM) devices based behavioral model for domain-wall nanowires. Unlike the current-controlled and voltage-controlled devices, the control in MCM device needs to be triggered by rising edge of one shift (SHF) signal, which can be formulated as

$$\theta = f\left(T_{sl}, \theta_r, T_{sr}, \theta_l, \theta_c\right) = T_{sl}\theta_r + T_{sr}\theta_l + \overline{T}_{sl}\overline{T}_{sr}\theta_0. \tag{5}$$

In which T_{sl} and T_{sr} are the shift-left and shift-right commands; θ_r and θ_l are the magnetization angles in right adjacent cell and left adjacent cell respectively; θ_c is the current state before the trigger signal. This describes that the θ-state will change when triggered and will remain state if no shift-signal is issued.

For the bit in MTJ, the applied voltage for spin-based read and write will also determine the θ-state as discussed previously. Therefore we have,

$$\theta = f(T_{sl}, \theta_r, T_{sr}, \theta_l, \theta_0) + g(Vp, Vn, \theta_c) \tag{6}$$

where Vp and Vn are the MTJ positive and negative nodal voltages, and $g(Vp, Vn, \theta_0)$ is the additional term that combines Equation (1) to (4).

In addition, the domain-wall propagation velocity can be mimicked by the SHF-request frequency. The link between the SHF-request frequency and the propagation velocity is the experimentally observed by current-velocity relation [22],

$$v = k(J - J_0), \tag{7}$$

where J is the injected current density and J_0 is the critical current density.

By combining equations (1) to (6) together, with the magnetization angles θ and ϕ as internal state variables other than electrical voltages and currents, one can fully describe the behaviors of the domain-wall nanowire device, where each domain is modeled as the proposed MCM device. As such, the modified nodal analysis (MNA) can be built in the SPICE-like simulator

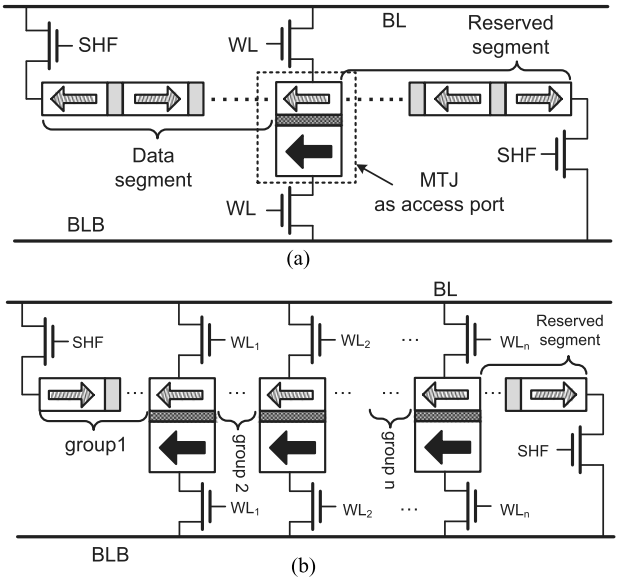


Fig. 4. Macro-cell of DWM with: (a) single access-port; and (b) multiple access-ports.

[23], [24] to verify circuit designs by domain-wall nanowire devices.

IV. Domain-Wall Nanowire Based Main Memory

Compared with the conventional SRAM or DRAM by CMOS, the domain-wall nanowire based memory (DWM) can demonstrate two major advantages. Firstly, extremely high integration density can be achieved since multiple bits can be packed in one macro-cell. Secondly, significant standby power reduction can be expected as a non-volatile device does not require to be powered to retain the stored data. In this section, we will present DWM-based design with macro-cell memory: structure, modeling, and data organization.

A. Domain-Wall Memory Macro-Cell Design

Fig. 4(a) shows the design of domain-wall nanowire based memory (DWM) macro-cell with access transistors. The access-port lies in the middle of the nanowire, which divides the nanowire into two segments. The left-half segment of nanowire is used for data storage while the right-half segment is reserved for shift-operation in order to avoid information lost. For the worst case scenario, in order to access the left-most bit of data segment, all information in data segment is shift to reserved segment with first bit aligned with access port. Without reserved segment the magnetization will move beyond the physical boundary and data will get lost. The idea behind is to provide temporary room, while the data in reserved segment are not important. In the worst case scenario discussed above, the reserved segment has to be at least as long as data segment. In such case, the data utilization rate is only 50%. In order to improve the data utilization rate, a multiple port macro-cell structure is presented in Fig. 4(b). The access-ports are equally distributed along the nanowire, which divides the nanowire into multiple segments.

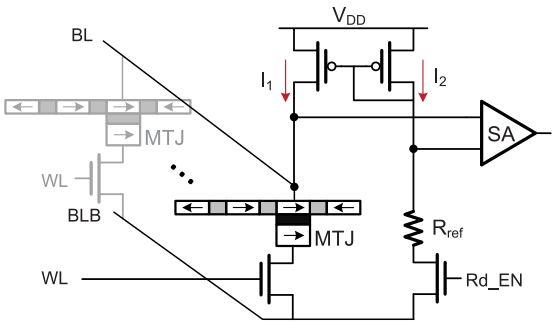


Fig. 5. Domain-wall memory read sensing circuit.

Except the right-most segment, all other segments are data segments with the bits in one segment form a $group$. In such case, to access arbitrary bit in the nanowire, the shift-offset is always less than the length of one segment, thus the data access can be done faster. As a moderate number of access-ports will be helpful to increase the data utilization rate, too many access-ports may lead to the opposite. This is because additional access-ports will incur area overhead to accommodate its access transistors.

The common read sensing circuit for DWM is shown in Fig. 5. To differentiate the high/low resistance values of MTJ in DWM, the reference cell has the resistance of $R_{\rm AP}||R_P$ [25]. During the read operation, the $\rm Rd\_EN$ is enabled, and bit-line (BL) is applied with read voltage that is less than threshold voltage for write operation. The current of MTJ branch will be mirrored to the reference cell branch, and the according voltages are compared by the sense amplifier (SA), which are two cross-coupled inverters. The write is performed by controlling polarity of write voltage between BL and BLB and asserting the according WL. In particular, when BL is high and BLB low, parallel or low-resistance state will be written to the MTJ; on the other hand, BL low and BLB high will result in anti-parallel or high-resistance state of MTJ. The selection of both cell and access port is achieved through word-line (WL). The number of sensing circuit for DWM depends on the width of bits to output, and sensing circuits are shared among different bit-lines by column multiplex [13], [26], [27].

The number of bits in one macro-cell can be calculated by

$$N_{\rm cell-bits} = (N_{\rm rw-ports} + 1)N_{\rm group-bits} \tag{8}$$

In which $N_{\rm rw-ports}$ is the number of access ports. Then the macro-cell area can be calculated by

$$A_{\rm nanowire} = N_{\rm cell-bits} L_{\rm bit} W_{\rm nanowire} \tag{9}$$

$$A_{cell} = A_{\rm nanowire} + 2A_{shf-nmos} + 2A_{rw-nmos} N_{\rm rw-ports} \tag{10}$$

where $L_{\rm bit}$ is the pitch size between two consecutive bits, $W_{\rm nanowire}$ the width of domain-wall nanowire, $A_{\rm shf-nmos}$ and $A_{\rm rw-nmos}$ are the transistor size at shift-port and access-port respectively.

Moreover, the bit-line capacitance is crucial in the calculation of latency and dynamic power. The increased bit-line

capacitance due to the multiple access-ports can be obtained by

$$C_{\mathrm{bit-line}} = (N_{\mathrm{rw-ports}} C_{\mathrm{drain-rw}} + C_{\mathrm{drain-shf}} + C_{\mathrm{bl-metal}}) \times N_{\mathrm{row}} \quad (11)$$

in which $C_{\mathrm{bl-metal}}$ is the capacitance of bit-line metal wire per cell, the $C_{\mathrm{drain-rw}}$ and $C_{\mathrm{drain-shf}}$ are the access-port and shift-port transistor drain capacitances, respectively. Note that the undesired increase of per-cell capacitance will be suppressed by the reduced number of rows due to higher nanowire utilization rate.

Besides the latency and energy on bit-line and sensing circuit, the domain-wall nanowire specific behaviors will also incur in-cell delay and energy dissipation. The magnetization reversal energy 0.27pJ and delay 600ps can be obtained through the transient analysis by the SPICE-like simulation as discussed in Section III. The read-energy is a few fJ. Also, the read-operation will not contribute in-cell delay. The delay of shift-operation can be calculated by

$$T_{\mathrm{shift}} = L_{\mathrm{bit}} / v_{\mathrm{prop}} \quad (12)$$

in which v_{prop} is the domain-wall propagation velocity that can be calculated by Equation (7). The Joule heat caused by the injected current is calculated as the shift-operation dynamic energy.

B. Cluster-Group Data Organization

There are two potential problems for the DWM macro-cell. Firstly, there exists variable access latencies for the bits that locate at different positions in the nanowire. Secondly, if the required bits are all stored in the same nanowire, very long access latency will be incurred due to the sequential access.

It is important to note that the data exchange between main memory and cache is always in the unit of a cache-line size of data, i.e. the main memory will be read-accessed when last-level cache miss occurs; and will be write-accessed when a cache-line needs to be evicted. Therefore, instead of the per access latency, the latency of the data block in the size of a cache-line becomes the main concern. Based on such fact, we present a cluster-group based data organization. The idea behind cluster is to distribute data in different nanowires thus they can be accessed in parallel to avoid the sequential access; and the idea behind group is to discard the within-group addressing, and transfer the $N_{\mathrm{group-bits}}$ bits in $N_{\mathrm{group-bits}}$ consecutive cycles, to avoid the variable latency. Specifically, a cluster is the bundle of domain-wall nanowires that can be selected together through bit-line multiplexers. The number of nanowires in one cluster equals the I/O bus bandwidth of the memory. Note that the data in one cache-line have consecutive addresses. Thus, by distributing the bits of N consecutive bytes, where N is decided by the I/O bus bandwidth, into different nanowire within a cluster, the required N bytes can be accessed in parallel to avoid the sequential access. In addition, within each nanowire in the cluster, the data will be accessed in the unit of group, i.e. the bits in each group will be accessed in consecutive cycles with a similar fashion as DRAM.

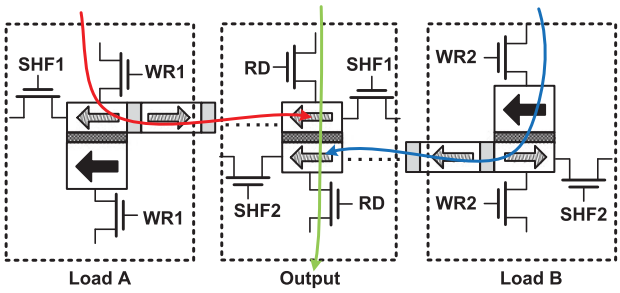


Fig. 6. Low power XOR-logic implemented by two domain-wall nanowires.

The number of groups per nanowire is thus decided by

$$N_{\mathrm{group-bits}} = N_{\mathrm{line-bits}} / N_{\mathrm{bus-bits}}. \quad (13)$$

For example, in system with cache-line size of 64-byte, and memory I/O bus bandwidth of 64-bit, the group size is 8-bit. As such, the DWM with cluster-group based data organization can be operated in the following steps:

- Step1: The group-head initially is aligned with the access-port, thus the distributed first 8 consecutive bytes can be first transferred between memory and cache;
- Step2: Shift the nanowire with 1-bit offset, and transfer the following 8 consecutive bytes. Iterate this step 6 more times until the whole cache-line data is transferred;
- Step3: After the data transfer is completed, the group-head is relocated to the initial position as required in step 1.

As mentioned in Section III-C, the current-controlled domain-wall propagation velocity is proportional to the applied shift-current,. By applying a larger shift-current, a fast one-cycle cluster head relocation can be achieved. In such a manner, the data-transfer of cache block will be able to achieve a fixed and also lowest possible latency.

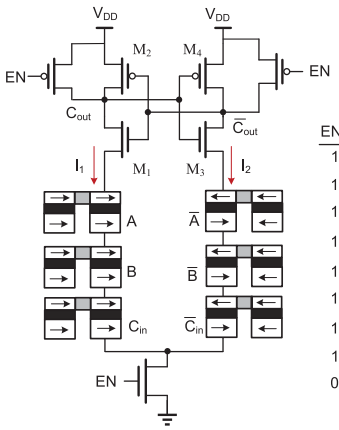
V. Domain-Wall Nanowire Based Logic

The magnetization switching with sub-nanosecond speed and and sub-pJ energy have been experimentally demonstrated [8]–[10]. As such, the domain-wall logic can be further explored for logic-in-memory based computing. In this section, we show how to further build domain-wall XOR logic, and how it is applied for low-power ALU design for comparison and addition operations.

A. Domain-Wall XOR Logic

The GMR-effect can be interpreted as the bitwise-XOR operation of the magnetization directions of two thin magnetic layers, where the output is denoted by high or low resistance. In a GMR-based MTJ structure, however, the XOR-logic will fail as there is only one operand as variable since the magnetization in fixed layer is constant. Nevertheless, this problem can be overcome by the unique domain-wall shift-operation in the domain-wall nanowire device, which enables the possibility of domain-wall XOR logic for computing.

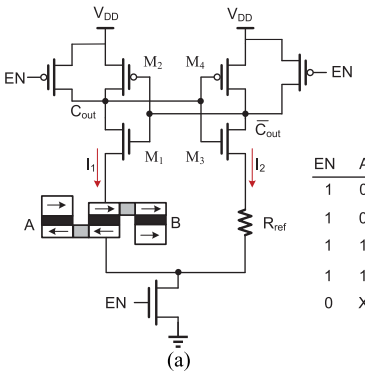
A bitwise-XOR logic implemented by two domain-wall nanowires is shown in Fig. 6. The proposed bitwise-XOR logic is performed by constructing a new read-only-port, where two free



EN	A	B	C	R_L>R_R?	C_{out}	$\overline{C}_{out}$
1	0	0	0	false	0	1
1	0	0	1	false	0	1
1	0	1	0	false	0	1
1	0	1	1	true	1	0
1	1	0	0	false	0	1
1	1	0	1	true	1	0
1	1	1	0	true	1	0
1	1	1	1	true	1	0
0	X	X	X	X	1	1

(b)

Fig. 7. Carry logic achieved by domain-wall nanowires.



EN	A	B	R_L	R_L>R_{ref}?	C_{out}	$\overline{C}_{out}$
1	0	0	P	false	0	1
1	0	1	AP	true	1	0
1	1	0	AP	true	1	0
1	1	1	P	false	0	1
0	X	X	X	X	1	1

(b)

Fig. 8. Sum logic achieved by domain-wall nanowires.

layers and one insulator layer are stacked. The two free layers are in the size of one magnetization domain and are from two respective nanowires. Thus, the two operands, denoted as the magnetization direction in free layer, can both be variables with values assigned through the MTJ of the according nanowire. As such, it can be shifted to the operating port such that the XOR-logic is performed.

For example, the $A \oplus B$ can be executed in the following steps

- The operands A and B are loaded into two nanowires by enabling WL_1 and WL_2 respectively;
- A and B are shifted from their access-ports to the read-only-ports by enabling SHF_1 and SHF_2 respectively;
- By enabling RD, the bitwise-XOR result can be obtained through the GMR-effect.

Note that in the x86 architecture processors, most XOR instructions also need a few cycles to load its operands before the logic is performed, unless the two operands are both in registers. As such, the proposed domain-wall XOR logic can be a potential substitution of the CMOS-based XOR-logic. Moreover, similar as the DWM macro-cell, zero leakage can be achieved for such XOR-logic.

B. Domain-Wall Adder

To realize a full adder, one needs both *sum* logic and *carry* logic. As the domain-wall nanowire based XOR logic has been achieved, the sum logic can be readily realized by deploying two units: Sum $= (A \oplus B) \oplus C$. As for carry logic, spintronics based carry operation is proposed in [28], where a pre-charge sensing amplifier (PCSA) is used for resistance comparison. The carry logic by PCSA and two branches of domain-wall nanowires is shown in Fig. 7(a). The three operands for carry operation are denoted by resistance of MTJ (low for 0 and high for 1), and belong to respective domain-wall nanowires in the left branch. The right branch is made complementary to the left one. Note that the C_{out} and $\overline{C_{\mathrm{out}}}$ will be pre-charged high at first when PCSA EN signal is low. The complementary values can be easily obtained by reversely placing the fixed layers of MTJs in the right branch. When the circuit is enabled, the branch with lower resistance will discharge its output to '0'. For example, when left branch has no or only one MTJ in high resistance, i.e. no carry out, the right branch will have three or two MTJs in high resistance, such that the C_{out} will be 0. The complete truth table is shown in Fig. 7(b), which is able to confirm carry logic by this circuit. The domain-wall nanowire works as the writing circuit for the operands by writing values at one end and shift it to PCSA. The *sum* logic of two operands by PCSA is shown in Fig. 8.

Together with CARRY logic, the domain-wall based half-adder can be implemented. By deploying two half-adders in series the domain-wall based full-adder can be achieved. Note that the domain-wall logic is sequential thus the full-adder is expected to have longer latency. We will show that the undesired long latency can be overcome by the MapReduce based matrix multiplication in Section VI-C.

C. Domain-Wall Multiplication

With the full adder implemented by domain-wall nanowires and intrinsic shift ability of domain-wall nanowire, the multiplication operation can be easily achieved by breaking it down to multiple domain-wall shift operations and additions. Operand A with m non-zero bits multiplied by operand B with n non-zero bits ($m > n$) can be decomposed into n shift operations and n additions. For example, multiplication of binary 1011 and 110 can be decomposed into addition of 10110 and 101100, where 10110 and 101100 are obtained by left-shifting 1011 one and two bits in domain-wall nanowire. As such, not only can the complicated domain-wall multiplier circuit be avoided, but also multiplication operation can be handled more efficiently by reusing domain-wall adders in a distributed MapReduce fashion, which will be discussed in Section VI.

D. Domain-Wall LUT Logic

Look-up table (LUT), essentially a pre-configured memory array, takes a binary address as input, finds target cells that contain result through decoders, and finally outputs correspondingly by sense amplifiers. A domain-wall nanowire based LUT is illustrated in Fig. 9. Compared with the conventional SRAM or DRAM by CMOS, the domain-wall LUT has higher density and lower leakage power.

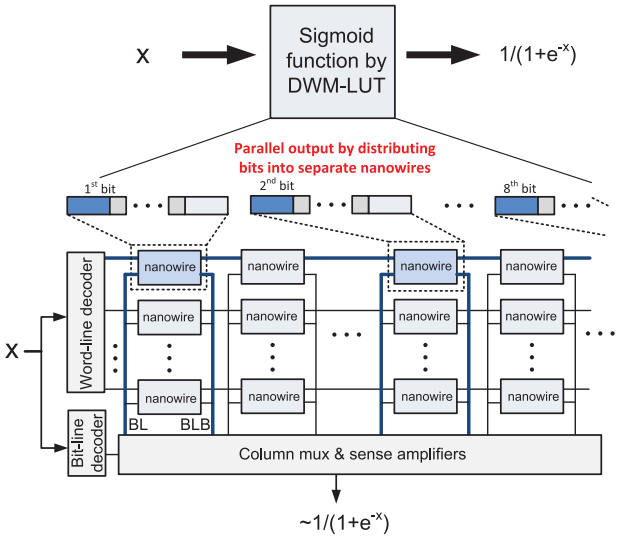


Fig. 9. Domain-wall LUT logic by domain-wall memory array.

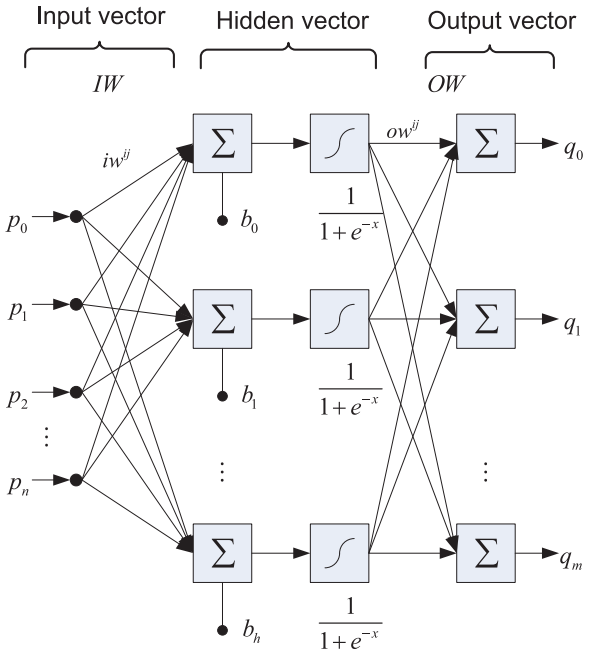


Fig. 10. Working flow of extreme learning machine.

Based on the way data is organized, the result can be output in sequential manner or parallel manner. In sequential output scenario, the binary result is stored in single domain-wall nanowire that is able to hold multiple bits of information. Assume each cell has only one access port and the first bit of result is initially aligned with access port, the way to output result is to iteratively readout and shift one bit until the last bit is output. In parallel output scenario, the multiple-bit result is distributed into different nanowires. Because each cell has their own access port, the multiple bits can be output concurrently. The design complexity of parallel output scheme is that, to find the relative position of the result within the nanowire, a variable access time will be introduced. For example, if the result is stored at first bit of the nanowires, the result can be readout in one cycle; on the contrary if the result is kept at the very last bit of the nanowires, it will take tens of cycles to shift first before the result is output. Therefore, the choice between sequential output and parallel output is the tradeoff between access latency and design complexity.

VI. In-Memory Machine Learning for Image Processing

In this section, we will use the extreme learning machine based super-resolution (ELM-SR) as a case study application, to show how machine learning algorithms can be efficiently executed within the proposed distributed in-memory computing platform. The application of ELM for image processing in this paper is an ELM based image super-resolution (SR) algorithm [29], which learns the image features of a specific category of images and improves low-resolution figures by applying learned knowledge. As demonstrated in Section V, we are able to achieve fundamental addition, multiplication, and LUT domain-wall logic. In the following, we will demonstrate how to map ELM-SR algorithm to the proposed distributed in-memory architecture.

A. Extreme Learning Machine

Among numerous machine learning algorithms [30]–[34], support vector machine (SVM) [30], [31] and neural network (NN) [32], [33] are widely discussed. However, both two algorithms have major challenging issues in terms of slow learning speed, trivial human intervene (parameter tuning) and poor computational scalability [34]. Extreme Learning Machine (ELM) was initially proposed [34], [35] for the single-hidden-layer feed-forward neural networks (SLFNs). Compared with traditional neural networks, ELM eliminates the need of parameter tuning in the training stage and hence reduces the training time significantly. The output function of ELM is formulated as (only one output node is considered)

$$f_L = \sum_{i=1}^{L} \beta_i h_i(X) = h(X)\beta \tag{14}$$

where $\beta = [\beta_1, \beta_2, \cdots, \beta_L]^T$ is the output weight vector storing the output weights between the hidden layer and output node. $h(X) = [h_1(X), h_2(X), \cdots, h_L(X)]^T$ is the hidden layer output matrix given input vector X and performs the transformation of input vector into L-dimensional feature space. The training process of ELM aims to obtain output weight vector β and minimize the training error as well as the norm of output weight

$$\text{Minimize} : \|H\beta - T\| \text{ and } \|\beta\| \tag{15}$$

$$\beta = H^{\dagger}T \tag{16}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H.

Note that ELM-SR is commonly used as pre-processing stage to improve image quality before applying other image algorithms. It involves intensive matrix operation, such as matrix addition, matrix multiplication as well as exponentiation on each element of a matrix. Fig. 10 illustrates the computation flow for ELM-SR, where input vector obtained from input image is multiplied by input weight matrix. The result is then added with

Algorithm 1 Matrix multiplication in MapReduce form

function Mapper($partitioned\ matrix\ p \in M, v$)
  for all $elements\ m_{ij} \in p$ **do**
    $b_{ijk} \leftarrow decompose(m_{ij}v_j)$
    $emit(i, b_{ijk})\ to\ list\ l_i$
  end for
end function
function Reducer(l_q)
  if $length\ of\ l_q > 1$ **then**
    $remove\ (q, v_1), (q, v_2)\ from\ list\ l_i$
    $emit(q, v_1 + v_2)\ to\ list\ l_i$
  end if
end function

bias vector b to generate input of sigmoid function. Lastly sigmoid function outputs are multiplied with output weight matrix to produce final results.

B. MapReduce-Based Matrix Multiplication

MapReduce [36] is a parallel programming model to efficiently handle large volume of data. The idea behind MapReduce is to break down large task into multiple sub-tasks, and each sub-task can be independently processed by different $Mapper$ computing units, where intermediate results are emitted. The intermediate results are then merged together to form the global results of the original task by the $Reducer$ computing units.

The problem to solve is $x = M \times v$. Suppose M is an $n \times n$ matrix, with element in row i and column j denoted by m_{ij}, and v is a vector with length of n. Hence, the product vector x also has the length of n, and can be calculated by

$$x_i = \sum_{j=1}^{n} m_{ij}v_j = \sum_{j=1}^{n}\sum_{k=1}^{l} b_{ijk}$$

where the multiplication of $m_{ij}v_j$ is decomposed into the sum of b_{ijk}. As such, the matrix multiplication can be purely calculated by addition operations, and thus the domain-wall adder logic can be exploited.

The pseudo-code of matrix multiplication in MapReduce form is demonstrated in Algorithm 1. Matrix M is partitioned into many blocks, and each Mapper function will take the entire vector v and one block of matrix M. For each matrix element m_{ij} it decomposes multiplication of $m_{ij}v_j$ into additions of multiple b_{ijk} and emits the key-value pair (i, b_{ijk}). The sum of all the values with same key i will make up the matrix-vector product element x_i. A reducer function simply has to sum all the values associated with a given key i. The summation process can be executed concurrently by iteratively summing two values and emitting one result until only one key-value pair is left for each key, namely the (i, x_i).

C. Workload Mapping

Fig. 11 shows how the MapReduce based ELM-SR is mapped into the proposed NVM based computing platform. It is one specific implementation of the local in-memory logic and data

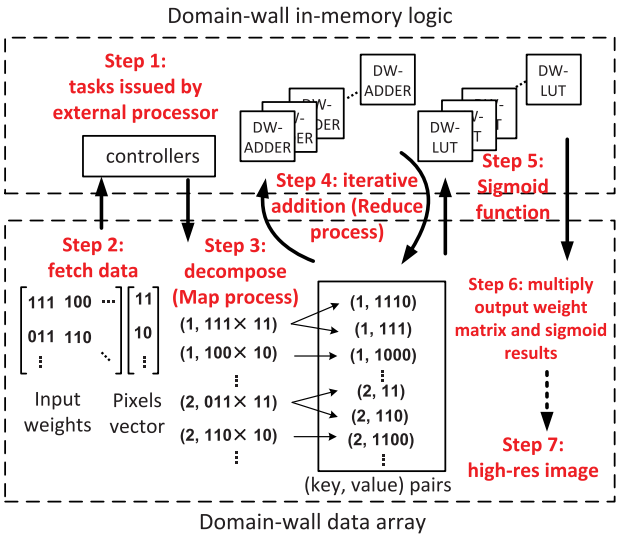


Fig. 11. ELM-SR algorithm mapping to proposed domain-wall nanowire based computing platform.

pair as shown in Fig. 2. The execution starts with a command issued by external processor to the memory. The local controller in the in-memory logic part, a simple state machine for example, then loads data from the data array: the randomized first-layer input weight matrix M and vector v that represents the low-resolution image. A *map* process follows to decompose the multiplications into multiple values to sum by domain-wall shift operations, and then emit <key, value> pairs accordingly. All emitted pairs are stored in a separate segment of data array called intermediate results pool.

The <key, value> pairs are further combined in the *reduce* process. Specifically, the controller will fetch elements in the intermediate results pool and dispatch them to available reducers, namely domain-wall adders as introduced in Section V. Each reducer will take two values with same key, combine the values by addition, and emit a new pair to the intermediate results pool. The reduce process works in an iterative manner, combining two pairs to one pair until the intermediate results can not be further combined. Instead of the single addition latency of domain-wall adder, the parallelism of additions in MapReduce fashion has direct impact on the latency of obtaining weighted sum results, especially in the data-intensive machine learning application. Therefore, though the domain-wall based adder needs multiple cycles to execute, this disadvantage can be suppressed in the MapReduce matrix multiplication.

The domain-wall LUTs, configured to execute Sigmoid function, are used to form the hidden layer vector. The hidden layer vector then multiplied by the output weight matrix, which is omitted in Fig. 11 for simplicity of illutration, as it is a repetition of step 2-4. Sigmoid function includes exponentiation, division, and addition, which is a computing-intensive operation in ELM application. In particular, the exponentiation will take many cycles to execute in the conventional processor due to the lack of corresponding accelerator. Therefore, it is extremely economic to perform exponentiation by look-up table. Note that the LUT size is determined by the input domain, the output

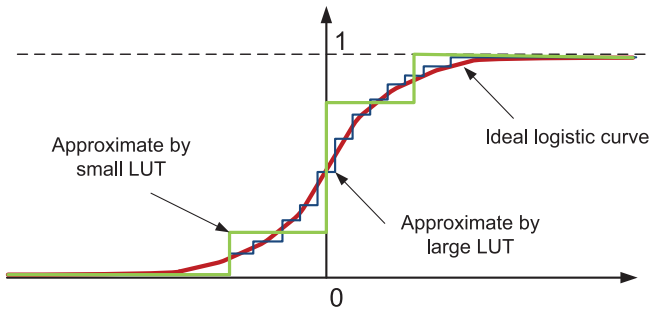


Fig. 12. Domain-wall LUT size effect on the precision of the sigmoid function. The larger the LUT, the smoother and more precise the curve is.

range, and the required precision for the floating point numbers. Fig. 12 shows the ideal logistic curve and approximated curves by LUTs. It can be observed that the output range is bounded between 0 and 1, and although the input domain is infinite, it is only informative in the center around 0. The LUT visually is the digitalized logistic curve and the granularity, i.e. precision, depends on the LUT size. For machine learning application, the precision is not as sensitive as scientific computations. As a result, the LUT size for sigmoid function can be greatly optimized and leads to high energy efficiency for sigmoid function execution. As such, the final results are obtained, and the results write back signifies the end of whole process.

VII. Experiment and Discussion

A. Experiment Setup

The experiment is performed within developed self-consistent simulation platform consisting device level NVM SPICE simulator NVM-SPICE [37], [38], system level memory evaluation tool CACTI [26], and system level processor power evaluating tool McPAT [39], and cycle-accurate architecture simulator gem5 [40]. For NVM-SPICE, the domain-wall nanowire device model introduced in Section III is implemented within NVM-SPICE, transient analysis of all domain-wall based circuits can be performed. As such, accurate operation energy and timing for domain-wall nanowire device and logic can be obtained. Both CACTI and McPAT are extended with domain-wall memory and logic model based on Section IV and V. The self-consistent simulation platform has the following setup for domain-wall nanowire: the technology node of 32 nm is assumed with width of 32 nm, length of 64 nm per domain, and thickness of 2.2 nm for one domain-wall nanowire; the permittivity of MgO layer is $8.8\epsilon_0$ [41], which leads to parasitic capacitance of 7.25e-17 F for MTJ; the Roff is set at 2600 Ω, the Ron at 1000 Ω, the writing current at 100 μA, and the current density at 6×10^8 A/cm^2 for shift-operation. As domain-wall logic compares the resistance of its two branches, the length of 64 nm is used for transistors to get higher output swing. For domain-wall memory, the default 32 nm length is used. For all transistors, W/L ratio is 1 and W_p/W_n ratio is 2.

B. Device and Circuit Level Evaluation of Domain-Wall Logic

1) XOR/Sum Logic: Fig. 13 shows both controlling timing diagram and operation details of domain-wall XOR logic,

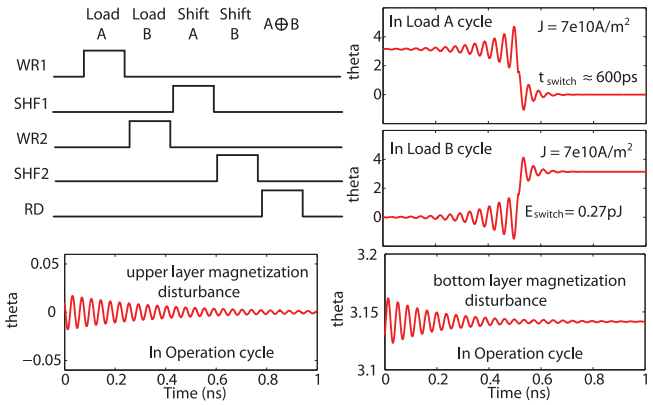


Fig. 13. Timing diagram of domain-wall XOR with NVM-SPICE simulation for each operation.

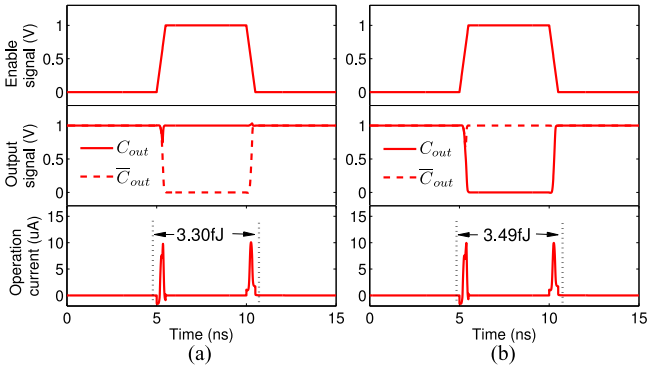


Fig. 14. NVM-SPICE simulation results for carry logic (a) A = 1, B = 1, and $C_{\rm in}$=1; (b) A = 0, B = 1, and $C_{\rm in}$=0.

which is simulated within the NVM-SPICE. The θ states of the nanowire that takes A are all initialized at 0, and the one takes B all at π. Only two-bit per nanowire is assumed for both nanowires. The operating-port is implemented as a developed magnetization controlled magnetization (MCM) device, with internal state variables θ and ϕ for both upper layer and bottom layer. In the cycles of $loadA$ and $loadB$, the precession switching can be observed for the MTJs of both nanowires. Also, the switching energy and time have been calculated as 0.27 pJ and 600 ps, which is consistent with the reported devices [8]–[10]. In the $shift$ cycles, triggered by the SHF-control signal, the dynamics θ and ϕ of both upper and bottom layers are updated immediately. In the $operation$ cycle, a subtle sensing current is applied to provoke GMR-effect. Subtle magnetization disturbance is also observed in both layers in the MCM device, which validates the read-operation. The θ values that differ from initial values in the $operation$ cycle also validate the successful domain-wall shift.

2) Carry Logic: As the domain-wall *carry* logic is symmetric, there are only two possible input scenarios, which are both simulated by NVM-SPICE, and the simulation results are shown in Fig. 14(a) and (b). For the scenario in Fig. 14(a), all three MTJs in the left branch are at anti-parallel states with high resistance, and their complementary MTJs in the right branch are at parallel states with low resistance. Before logic is enabled, both $C_{\rm out}$ and $\overline{C}_{\rm out}$ are logical high, and when the

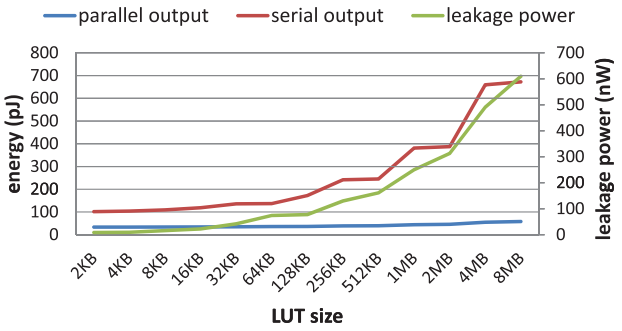


Fig. 15. Power characterization for domain-wall LUT in different sizes.

enable signal is asserted, the $\overline{C}_{\rm out}$ which represents the branch with lower resistance is pulled down quickly, as expected. Similarly for scenario in Fig. 14(b), where one MTJ state different from the other two, the $C_{\rm out}$ that represents the branch with lower resistance is pulled down to logical low after enable signal is asserted. All other input combinations are equivalent to either case therefore the *carry* logic can be validated. For both case, the operation current peaked at 10 μA, which is far less than the writing current 100 μA, and thus will not accidentally switch input operands and lead to incorrect result. By integrating the current-voltage product within the marked range, the energy consumption in this step can be calculated to be 3.3/3.49 fJ.

3) LUT: Fig. 15 shows the power characterization of domain-wall LUT in different array sizes. In terms of dynamic energy per look-up operation, the parallel output scenario is much more power efficient than serial output scenario, and the gap enlarges when array size increases. This is because more cycles are required to output results in serial than in parallel, therefore more access operations are involved. However, the serial scenario is able to avoid the variable access latency issue, which reduces the design complexity of the controller. For leakage power, the non-volatility leads to extremely low leakage power in nW scale, which is negligible compared with its dynamic power. For volatile SRAM and DRAM, the leakage power may consume as large as half the total power especially in advanced technology node [26].

Once the domain, range, and precision of function are decided, the domain-wall LUT size can be determined accordingly. Therefore, the power characterization can be used as a quick reference to estimate the power profile of specific function to perform system level design exploration and performance evaluation.

C. System Level Evaluation of Domain-Wall Logic and Memory

For machine learning applications, besides the core matrix multiplication there are still many instructions that cannot be executed by MapReduce based in-memory domain-wall logic. In this part, both the domain-wall nanowire based external processor and memory (as shown in Fig. 2) are evaluated. The core and memory configurations are shown in Table I. The conventional logic in external processor is replaced by their domain-wall logic and memory counterparts. The 32-bit 65nm processor

TABLE I
EXTERNAL PROCESSOR AND MEMORY CONFIGURATIONS FOR REAL CASE DOMAIN-WALL LOGIC AND MEMORY EVALUATION

Processor	
Number of cores	4
Frequency	1 GHz
Architecture	x86, O3, issue width - 4, 32 bits
Functional units	Integer ALU - 6 Complex ALU - 1 Floating point unit - 2
Cache	L1: 32 KB - 8 way/32 KB - 8 way L2: 1 MB - 8 way Line size - 64 bytes
Memory	
Technology node	32 nm
Memory size	2 GB - 128 MB per bank
IO bus width	64 bits

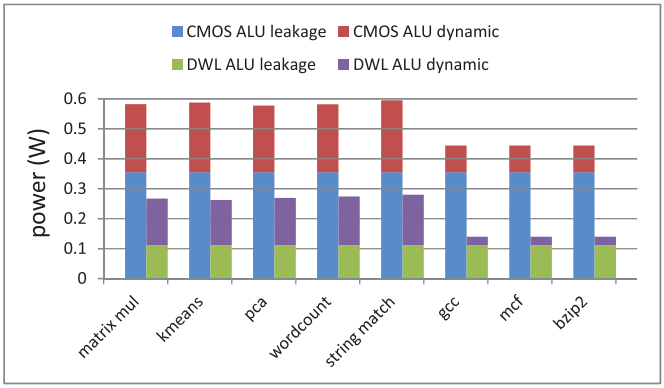


Fig. 16. Per core ALU power comparison between CMOS design and domain-wall logic based design.

is assumed with four cores integrated. In each core, there are 6 integer ALUs which executes XOR, OR, AND, NOT, ADD and SUB operations, and complex integer operations like MUL, DIV are executed in integer MUL. The 32nm technology node and 64-bit I/O bus width are assumed for memory.

1) Domain-Wall Logic: For domain-wall logic based ALU design evaluation, firstly the gem5 simulator is employed to take both SPEC2000 and Phoenix benchmarks [42] and to generate the runtime instruction and memory accessing traces. The trace file is then analyzed with the statistics of instructions that can be executed on the proposed XOR and adder for logic evaluation. The L2-cache-miss rates are also generated, in order to obtain the actual memory access for memory power evaluation. Then, McPAT is modified to evaluate power of the 32-bit ALU that is able to perform XOR, OR, AND, NOT, ADD and SUB operations. The instruction controlling decoder circuit is also considered during the power evaluation. The leakage power of both designs is calculated at gate level by the McPAT power model.

Fig. 16 presents the per-core ALU power comparison between the conventional CMOS design and domain-wall logic based design. Benefited from the use of domain-wall logic, both of the dynamic power and leakage power can be greatly reduced. It can

TABLE II
PERFORMANCE COMPARISON OF 128 MB MEMORY-BANK IMPLEMENTED BY DIFFERENT STRUCTURES

Memory structure	area (mm^2)	access energy (nJ)	access time (ns)	leakage (mW)
DRAM	20.5	0.77	3.46	620.2
DWM/1 port	8.9	0.65	1.90	48.4
DWM/2 ports	6.2	0.72	1.71	30.1
DWM/4 ports	6.2	0.89	1.69	24.3
DWM/8 ports	5.7	1.31	1.88	19.0

be observed that the set of Phoenix benchmarks consume higher dynamic power compared to those of SPEC2006, which is due to the high parallelism of MapReduce framework with high utilization rate of the ALUs. Among each set, the power results exhibit a low sensitivity to the input, which indicates that percentages of instructions executed in XOR and ADDER of ALU are relatively stable even for different benchmarks. The stable improvement ensures the extension of the proposed domain-wall logic to other applications. Averagely, a dynamic power reduction of 31% and leakage power reduction of 65% can be achieved for ALU logic based on all the eight benchmarks.

2) Domain-Wall Memory: Table II shows the 128 MB memory-bank comparison between CMOS-based memory (or DRAM) and domain-wall nanowire based memory (or DWM). The number of access ports in main memory is varied for design exploration. The results of DRAM are generated by configuring the original CACTI with 32 nm technology node, 64-bit of I/O bus width with leakage optimized. The results of the DWM are obtained by the modified CACTI according to Section IV with the same configuration.

It can be observed that the memory area is greatly reduced in the DWM designs. Specifically, the DWMs with 1/2/4/8 access ports can achieve the area saving of 57%,70%,70% and 72%, respectively. The trend also indicates that the increase of number of access-ports will lead to higher area saving. This is because of the higher nanowire utilization rate, and is consistent with the analysis discussed in Section IV. Note that the area saving in turn results in a smaller access latency, and hence the DWM designs on average provide 1.9X improvement on the access latency. However, the DWM needs one more cycle to perform shift operation, which will cancel out the latency advantage. Overall, the DWM and DRAM have similar speed performance. In terms of power, the DWM designs also exhibit benefit with significantly leakage power reduction. The designs with 1/2/4/8 access ports can achieve 92%,95%,96% and 97% leakage power reduction rates, respectively. The advantage mainly comes from the non-volatility of domain-wall nanowire based memory cells. The reduction in area and decoding peripheral circuits can further help leakage power reduction in DWM designs. In addition, the DWM designs have the following trend of access energy when increasing the number of access ports. The designs with 1/2 ports require 16% and 6% less energy, while designs with 4/8 ports incur 15% and 70% higher access energy cost. This is because when the number of ports increases, there are more

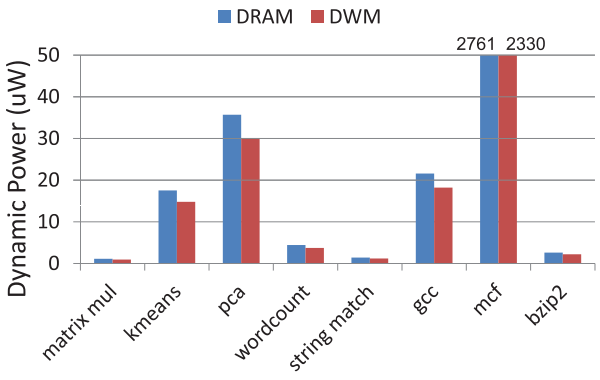


Fig. 17. (a) Runtime dynamic power of both DRAM and DWM under Phoenix and SPEC2006 (b) the normalized intended memory accesses.

TABLE III
AREA, POWER, THROUGHPUT AND ENERGY EFFICIENCY COMPARISON BETWEEN IN-MEMORY ARCHITECTURE AND CONVENTIONAL ARCHITECTURE FOR ELM-SR

Platform	Proposed	GPP (on-chip memory)	GPP (off-chip memory)
# of logic units	1 × processor 7714 × DW-ADDER 551 × DW-LUT 1 × controller	1 × processor	1 × processor
Logic Area (mm^2)	18 (processor) + 0.5 (accelerators)	18	18
Logic Power (Watt)	10.1	12.5	12.5
Throughput (MBytes/s)	108	9.3	9.3
EPB (nJ)	7	total: 394 I/O: 364 (92%) logic: 30 (8%)	total: 4127 I/O: 4097 (99%) logic: 30 (1%)

transistors connected to the bit-line which leads to increased bit-line capacitance.

The runtime dynamic power comparison under different benchmark programs are shown in Fig. 17(a). It can be seen that the dynamic power is very sensitive to the input benchmark, and the results of the Phoenix benchmarks shows no significant difference from those in SPEC2006. This is because the dynamic power is effected by both intended memory access frequency and the cache miss rate. Although the data-driven Phoenix benchmarks have much higher intended memory reference rate, both L1 and L2 cache miss rates of Phoenix benchmarks are much lower than SPEC2006, which is due to the very predictable memory access pattern when exhaustively handling data in Phoenix benchmarks. Overall, the low cache miss rates of Phoenix benchmarks cancel out the higher memory reference demands, which leads to a modest dynamic power. Also, the runtime dynamic power contributes much less to the total power consumption compared to leakage power, thus the leakage reduction should be the main design objective when determining the number of access ports.

(a) (b) (c)

Fig. 18. (a) Original image before ELM-SR algorithm (SSIM value is 0.91); (b) Image quality improved after ELM-SR algorithm by DW-NN hardware implementation (SSIM value is 0.94); (c) Image quality improved by GPP platform (SSIM value is 0.97).

D. Architecture Level Evaluation of Distributed In-Memory Computing

In this part, we will show the throughput and energy efficiency improvement brought by proposed purely domain-wall nanowire based distributed in-memory computing architecture. As a case study, the data-driven extreme learning machine based super-resolution (ELM-SR) application is executed within the proposed architecture, as discussed in Section VI. We will compare the proposed in-memory platform with the conventional general purpose processor (GPP) based platform. The evaluation of ELM-SR in GPP platform is based on gem5 and McPAT for core power and area model. Proposed in-memory computing architecture is evaluated in our developed self-consistent simulation platform based on NVMSPICE, DW-CACTI, and ELM-SR behavioral simulator. The processor runs at 3 GHz while the accelerators run at 500 MHz. System memory capacity is set as 1GB, and bus width is set as 128 bits. Based on most recent on-chip interconnect and PCB interconnect studies in [43], [44], 40 fJ/bit/mm for on-chip interconnect and 30 pJ/bit/cm for PCB interconnect are used as I/O overhead. For core-memory distance, 10 mm is assumed for on-chip case and 10 cm is assumed for PCB trace length, both according to [43], [44].

Table III compares ELM-SR in both proposed in-memory computing platform and GPP platforms. Due to the deployment of in-memory accelerators and high data parallelism, the throughput of proposed in-memory computing platform improves by 11.6x compared to GPP platform. In terms of area used by computational resources, proposed in-memory computing platform is 2.7% higher than that of GPP platform. Additional 0.5 mm^2 is used to deploy the domain-wall nanowire based accelerators. Thanks to the high integration density of domain-wall nanowires, the numerous accelerators are brought with only slight area overhead. In proposed in-memory computing platform, the additional power consumed by accelerators is compensated by the saved dynamic power of processor, since the computation is mostly performed by the in-memory logic. Overall, proposed in-memory computing platform achieves a power reduction of 19%. The most noticeable advantage of proposed in-memory computing platform is its much higher energy efficiency, energy-per-bit (EPB) as metrics, compared to GPP. Specifically, it is 56x and 590x better than that of GPP with on-chip and off-chip memory respectively. The advantage comes from three aspects: (a) in-memory computing architecture that saves I/O overhead; (b) non-volatile domain-wall nanowire devices that are leakage free; and (c) application specific accelerators. Specifically, the use of domain-wall logic/accelerators contributes to 4x improvement, while the in-memory architecture contributes to the rest (save of I/O overhead).

Fig. 18 shows the image quality comparison between the proposed in-memory architecture hardware implementation and the conventional GPP software implementation. To measure the performance quantitatively, structural similarity (SSIM) [45] is used to measure image quality after ELM-SR algorithm. It can be observed that the images after ELM-SR algorithm in both platforms have higher image quality than the original low-resolution image. However, due to the use of LUT, which trades off precision against the hardware complexity, the image quality in DW-NN is slightly lower than that in GPP. Specifically, the SSIM is 0.94 for DW-NN, 3% lower than 0.97 for GPP.

VIII. Conclusion

This paper has presented the memory design and logic design both built by the newly introduced domain-wall nanowire. The experimental results show that the domain-wall memory can reduce 92% leakage power and 16% dynamic power compared to main memory implemented by DRAM; and domain-wall logic can reduce 31% both dynamic and 65% leakage power under the similar performance compared to CMOS transistor based logic. And with the domain-wall memory and domain-wall logic, we further proposed a distributed in-memory architecture which is able to significantly improve the system throughput as well as energy efficiency. We show that the I/O traffic in the proposed DW-NN is greatly alleviated with an energy efficiency improvement by 56x and throughput improvement by 11.6x compared to the conventional image processing system by general purpose processor.

References

[1] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proc. IEEE*, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[2] A. Agarwal, S. Mukhopadhyay, C. H. Kim, A. Raychowdhury, and K. Roy, "Leakage power analysis and reduction for nano-scale circuits," *IEEE Micro*, vol. 26, no. 2, pp. 68–80, March 2006.

[3] H.-S. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.

[4] B. C. Lee, P. Zhou, J. Yang, Y. Zhang, B. Zhao, E. Ipek, O. Mutlu, and D. Burger, "Phase-change technology and the future of main memory," *IEEE Micro*, vol. 30, no. 1, p. 143, Jan. 2010.

[5] K. Tsuchida, T. Inaba, K. Fujita, Y. Ueda, T. Shimizu, Y. Asao, T. Kajiyama, M. Iwayama, K. Sugiura, S. Ikegawa, T. Kishi, T. Kai, M. Amano, N. Shimomura, H. Yoda, and Y. Watanabe, "A 64mb mram with clamped-reference and adequate-reference schemes," in *Proc. Solid-State Circuits Conf. Digest Tech. Papers*, Feb. 2010, pp. 258–259.

[6] Y. Wang, H. Yu, and W. Zhang, "Nonvolatile cbram-crossbar-based 3-d-integrated hybrid memory for data retention," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 22, no. 5, pp. 957–970, May 2014.

[7] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[8] H. Zhao, B. Glass, P. K. Amiri, A. Lyle, Y. Zhang, Y.-J. Chen, G. Rowlands, P. Upadhyaya, Z. Zeng, J. A. Katine, J. Langer, K. Galatsis, H. Jiang, K. L. Wang, I. N. Krivorotov, and J.-P. Wang, "Sub-200 ps spin transfer torque switching in in-plane magnetic tunnel junctions with interface perpendicular anisotropy," *J. Phys. D: Appl. Phys.*, vol. 45, no. 2, pp. 025001-1–025001-4, 2012.

[9] G. E. Rowlands, T. Rahman, J. A. Katine, J. Langer, A. Lyle, H. Zhao, J. G. Alzate, A. A. Kovalev, Y. Tserkovnyak, Z. M. Zeng, H. W. Jiang, K. Galatsis, Y. M. Huai, P. K. Amiri, K. L. Wang, I. N. Krivorotov, and J.-P. Wang, "Deep subnanosecond spin torque switching in magnetic tunnel junctions with combined in-plane and perpendicular polarizers," *Appl. Phys. Lett.*, vol. 98, no. 10, pp. 102509-1–102509-3, 2011.

[10] H. Zhao, A. Lyle, Y. Zhang, P. K. Amiri, G. Rowlands, Z. Zeng, J. Katine, H. Jiang, K. Galatsis, K. L. Wang, I. N. Krivorotov, and J.-P. Wang, "Low writing energy and sub nanosecond spin torque transfer switching of in-plane magnetic tunnel junction for spin torque transfer random access memory," *J. Appl. Phys.*, vol. 109, no. 7, pp. 07C720-1–07C720-3, 2011.

[11] S. S. P. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, 2008.

[12] L. Thomas, S.-H. Yang, K.-S. Ryu, B. Hughes, C. Rettner, D.-S. Wang, C.-H. Tsai, K.-H. Shen, and S. S. Parkin, "Racetrack memory: A high-performance, low-cost, non-volatile memory based on magnetic domain walls," in *Proc. Electron. Devices Meeting,* 2011, pp. 24–2.

[13] R. Venkatesan, V. Kozhikkottu, C. Augustine, A. Raychowdhury, K. Roy, and A. Raghunathan, "Tapecache: A high density, energy efficient cache based on domain wall memory," in *Proc. IEEE Int. Symp. Low Power Electron. Design*, 2012, pp. 185–190.

[14] S. Matsunaga, J. Hayakawa, S. Ikeda, K. Miura, T. Endoh, H. Ohno, and T. Hanyu, "Mtj-based nonvolatile logic-in-memory circuit, future prospects and issues," in *Proc. Conf. Design, Autom. Test Europe*, 2009, pp. 433–435.

[15] S. Matsunaga, J. Hayakawa, S. Ikeda, K. Miura, H. Hasegawa, T. Endoh, H. Ohno, and T. Hanyu, "Fabrication of a nonvolatile full adder based on logic-in-memory architecture using magnetic tunnel junctions," *Appl. Phys. Exp.*, vol. 1, no. 9, pp. 091301-1–091301-3, 2008.

[16] W. H. Kautz, "Cellular logic-in-memory arrays," *IEEE Trans. Comput.*, vol. 100, no. 8, pp. 719–727, Aug. 1969.

[17] H. Kimura, T. Hanyu, M. Kameyama, Y. Fujimori, T. Nakamura, and H. Takasu, "Complementary ferroelectric-capacitor logic for low-power logic-in-memory vlsi," *IEEE J. Solid-State Circuits*, vol. 39, no. 6, pp. 919–926, Jun. 2004.

[18] S. Paul, S. Mukhopadhyay, and S. Bhunia, "A circuit and architecture codesign approach for a hybrid cmos–sttram nonvolatile fpga," *IEEE Trans. Nanotechnol.*, vol. 10, no. 3, pp. 385–394, May 2011.

[19] T. Hanyu, K. Teranishi, and M. Kameyama, "Multiple-valued logic-in-memory vlsi based on a floating-gate-mos pass-transistor network," in *Proc. Solid-State Circuits Conf.,*1998, pp. 194–195.

[20] Y. Wang, H. Yu, D. Sylvester, and P. Kong, "Energy efficient in-memory aes encryption based on nonvolatile domain-wall nanowire," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, 2014, pp. 1–4.

[21] M. D. Stiles and J. Miltat, "Spin-transfer torque and dynamics," in *Spin Dynamics in Confined Magnetic Structures III*. New York, NY, USA: Springer, 2006, pp. 225–308.

[22] D. Chiba, G. Yamada, T. Koyama, K. Ueda, H. Tanigawa, S. Fukami, T. Suzuki, N. Ohshima, N. Ishiwata, Y. Nakatani, and T. Ono, "Control of multiple magnetic domain walls by current in a co/ni nano-wire," *Appl. Phys. Exp.*, vol. 3, no. 7, pp. 073004-1–073004-3, 2010.

[23] W. Fei, H. Yu, W. Zhang, and K. S. Yeo, "Design exploration of hybrid cmos and memristor circuit by new modified nodal analysis," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 20, no. 6, pp. 1012–1025, Jun. 2012.

[24] H. Yu and Y. Wang, "Nonvolatile state identification and nvm spice," in *Design Exploration of Emerging Nano-Scale Non-Volatile Memory*. New York, NY, USA: Springer, 2014, pp. 45–83.

[25] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *Proc. High Performance Comput. Archit.*, 2011, pp. 50–61.

[26] S. J. Wilton and N. P. Jouppi, "Cacti: An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model," *IEEE J. Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, 1996.

[27] A. Iyengar and S. Ghosh, "Modeling and analysis of domain wall dynamics for robust and low-power embedded memory," in *Proc. Design Autom. Conf.*, 2014, pp. 1–6.

[28] H.-P. Trinh, W. Zhao, J.-O. Klein, Y. Zhang, D. Ravelsona, and C. Chappert, "Domain wall motion based magnetic adder," *Electron. Lett.*, vol. 48, no. 17, pp. 1049–1051, 2012.

[29] L. An and B. Bhanu, "Image super-resolution by extreme learning machine," in *Proc. Image Process.*, 2012, pp. 2209–2212.

[30] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[31] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[32] B. Yegnanarayana, *Artificial Neural Networks*. New Delhi, India: PHI Learning Pvt. Ltd., 2004.

[33] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesus, *Neural Network Design*. Boston, MA, USA: PWS, 1996.

[34] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[35] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proc. Neural Netw.,* 2004, pp. 985–990.

[36] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *ACM Commun.*, vol. 51, no. 1, pp. 107–113, 2008.

[37] (2012). Nvm-spice [Online]. Available: http://www.nvmspice.org/

[38] Y. Wang, W. Fei, and H. Yu, "Spice simulator for hybrid cmos memristor circuit and system," in *Proc. Cellular Nanoscale Netw. Appl.*, 2012, pp. 1–6.

[39] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. IEEE Int. Symp. Microarchit.*, 2009, pp. 469–480.

[40] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[41] P. Padhan, P. LeClair, A. Gupta, K. Tsunekawa, and D. Djayaprawira, "Frequency-dependent magnetoresistance and magnetocapacitance properties of magnetic tunnel junctions with mgo tunnel barrier," *Appl. Phys. Lett.*, vol. 90, no. 14, pp. 142105-1–142105-3, 2007.

[42] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multiprocessor systems," in *Proc. High Perform. Comput. Archit.*, 2007, pp. 13–24.

[43] S. Park, M. Qazi, L.-S. Peh, and A. P. Chandrakasan, "40.4 fj/bit/mm low-swing on-chip signaling with self-resetting logic repeaters embedded within a mesh noc in 45 nm soi cmos," in *Proc. Conf. Design, Autom. Test Europe*, 2013, pp. 1637–1642.

[44] V. Kumar, R. Sharma, E. Uzunlar, L. Zheng, R. Bashirullah, P. Kohl, M. S. Bakir, and A. Naeemi, "Airgap interconnects: Modeling, optimization, and benchmarking for backplane, pcb, and interposer applications," *IEEE Trans. Components, Packag. Manuf. Technol.*, vol. 4, no. 8, pp. 1335–1346, Aug. 2014.

[45] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

Yuhao Wang received the B.S. degree in microelectronics engineering from Xi'an Jiao Tong University, Xi'an, China, in 2011, and the Ph.D. degree in 2015 from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He is currently a Research Fellow at Nanyang Technological University. His research interests include emerging nonvolatile memory, in-memory architecture and big-data analytics.

Hao Yu (M'06–SM'14) received the B.S. degree from Fudan University, Shanghai, China, in 1999, and the M.S. and Ph.D. degrees from the Electrical Engineering Department, University of California, Los Angeles, CA, USA, in 2007, with major in integrated circuit and embedded computing. He was a Senior Research Staff at Berkeley Design Automation. Since October 2009, he has been an Assistant Professor at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His primary research interest is in emerging CMOS technologies such as 3DIC and RFIC designs at nanotera scale. He has 150 top-tier peer-reviewed publications, four books and five book chapters. He received the Best Paper Award from the ACM Transactions on Design Automation of Electronic Systems (TODAES) in 2010, Best Paper Award nominations in DAC06, ICCAD06, and ASP-DAC12, and Inventor Award from Semiconductor Research Cooperation. He is associate editor and technical program committee member of several journals and conferences.

Leibin Ni received the B.S. degree in microelectronics from Shanghai Jiao Tong University, Shanghai, China, in 2014. He is currently working toward the M.Eng. degree at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His current research interests include emerging nonvolatile memory platform and big-data in-memory computing.

Guang-Bin Huang (M'98–SM'04) received the B.Sc. degree in applied mathematics and the M.Eng. degree in computer engineering from Northeastern University, Shenyang, China, in 1991 and 1994, respectively, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 1999. During undergraduate period, he also concurrently studied at the Applied Mathematics Department and Wireless Communication Department, Northeastern University. From June 1998 to May 2001, he worked as a Research Fellow at the Singapore Institute of Manufacturing Technology (formerly known as the Gintic Institute of Manufacturing Technology) where he has led/implemented several key industrial projects. From May 2001, he has been working as an Assistant Professor and Associate Professor at the School of Electrical and Electronic Engineering, Nanyang Technological University. His current research interests include machine learning, computational intelligence, and extreme learning machines. He serves as an Associate Editor of *Neurocomputing* and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: PART B.

Mei Yan, Chuliang Weng, Wei Yang, and Junfeng Zhao, biographies not available at the time of publication.