Domain Wall Memory based Convolutional Neural Networks for Bit-width Extendability and Energy-Efficiency

Jinil Chung, Jongsun Park
School of Electrical Engineering, Korea University,
Seoul, Korea
{jinil_chung, jongsun}@korea.ac.kr

Swaroop Ghosh
Department of Computer Science and Engineering,
University of South Florida, Tampa, FL, USA
swaroopghosh@mail.usf.edu

ABSTRACT

In the hardware implementation of deep learning algorithms such as Convolutional Neural Networks (CNNs), vector-vector multiplications and memories for storing parameters take a significant portion of area and power consumption. In this paper, we propose a Domain Wall Memory (DWM) based design of CNN convolutional layer. In the proposed design, the resistive cell sensing mechanism is efficiently exploited to design a low-cost DWM-based cell arrays for storing parameters. The unique serial access mechanism and small footprint of DWM are also used to reduce the area and power cost of the input registers for aligning inputs. Contrary to the conventional implementation using Memristor-Based Crossbar (MBC), the bit-width of the proposed CNN convolutional layer is extendable for high resolution classifications and training. Simulation results using 65 nm CMOS process show that the proposed design archives 34% of energy savings compared to the conventional MBC based design approach.

CCS Concepts

• Hardware → Spintronics and magnetic technologies
• Hardware → Emerging architectures • Hardware → Application specific integrated circuits.

Keywords

Domain Wall Memory; Convolutional Neural Networks; Embedded Memory

1. INTRODUCTION

Convolutional Neural Networks (CNNs) are one of the well-known deep learning algorithms that have been widely employed for image classifications [1]. CNN can achieve low error rate with an acceptable complexity by combining three architectural ideas – local connections, shared weights, and spatial/temporal subsampling [2]. Deeper architecture or more layers in CNNs is the key to achieve high accuracy at the cost of complexity. In the hardware implementation of CNNs, vector-vector multiplications for extracting features and memories for storing parameters (weights and biases) take a significant portion of area and power consumption.

This work was supported by Semiconductor Research Corporation (#2442.001), National Science Foundation (CNS-1513126), and National Research Foundation of Korea (#2015M3D1A1070465).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ISLPED '16, August 08-10, 2016, San Francisco Airport, CA, USA

© 2016 ACM. ISBN 978-1-4503-4185-1/16/08 \$15.00

DOI: <http://dx.doi.org/10.1145/2934583.2934602>

Figure 1. Schematic of the DWM using the shift-based write scheme [6]. The MTJ at read head, extra two fixed layers at write head, and the overhead bits are also shown.

Emerging technologies such as Memristor-Based Crossbar (MBC) has been widely studied for area and power efficient implementation of neuromorphic computing [3-4]. In order to achieve a similar accuracy with the floating-point operation [5], the required bit-width of the parameters in fixed-point operation exceeds 7-8-bit according to the recent numerical results of Deep Neural Networks (DNNs) and/or CNNs. This is a critical programming limitation in a memristor device [7]. Without innovative memristor device improvements and analog circuit techniques, MBC-based design is hard to be extended to high accuracy applications. Hence it cannot replace the GPU-based systems as the training accelerator. In order to implement high accuracy training accelerator for image classification applications, bit-width extendability should be supported in CNN implementation with reasonable energy-efficiency.

DWM is a flavor of spintronic memory technology that possesses ability to store multiple bits per cell [8-9]. The bits are stored in the magnetic nanowire in the form of magnetic orientation and can be accessed serially by performing shift operations. DWM has been proposed for energy efficient in-memory machine learning [10]. In this work, the unique shift-based access pattern of DWM is efficiently exploited to implement the input registers of CNN convolutional layer that prefer sequential access. The resistor cell sensing circuit through read MTJ is also proposed to implement a partial dot product in CNN. Since the parameters are updated and stored in DWM with binary numbers in our design, the bit-width of the proposed CNN convolutional layer is easily extendable for high resolution training for image classifications. *To the best of our knowledge, this is the first study of the DWM based CNN convolutional layer implementation.* Particularly, we make the following contributions in this paper:

- We propose a novel DWM-based cell array architecture for area and energy-efficient CNN convolutional layer by partial implementation of a dot product using the resistor cell sensing circuit.

Table 1. Characteristics of DWM

Operation	Speed	Power	Footprint
Read	2.81 ns	23.08 μ W	2.56F ² per domain
Write	3.9 ns	55 μ W	(4 heads)
Shift	1 ns-2 ns	10 μ W	
Shift-based write	1 ns-2 ns	28.4 μ W	24.7F ² per bit

- We propose a novel DWM-based input register architecture by exploiting the sequential access pattern of DWM.

The rest of the paper is organized as follows. Section 2 describes the overview of basics of DWM. The details of a convolutional layer in CNNs are presented in Section 3. Section 4 provides the DWM-based CNN convolutional layer design approaches and implementations. Numerical results are presented in Section 5, and finally, conclusions are drawn in Section 6.

2. DWM FUNDAMENTALS

DWM consists of three components: (a) write head (b) read head and (c) magnetic Nanowire (NW). As shown in Figure 1, the read and write heads are similar to the conventional Magnetic Tunnel Junction (MTJ) whereas NW holds the bits in the form of magnetic polarity. Since a single bit-cell can hold multiple bits in the NW, this memory technology provides high-density. Magnetic NW is the crucial component that holds the bits. In essence, the NW is analogous to a shift register. The NW typically contains physical notches to move the DW in lockstep fashion [6,9,11]. It also ensures that the DW does not land in between two notches. The shift pulse is enough to dislodge the DW and shift along the NW. Note that the MTJ forms naturally between the NW and the fixed magnetic layer that are separated by the tunnel oxide barrier. The left (right) magnetic orientation in the NW can be regarded as ‘0’ (‘1’). The most interesting feature of the NW is the formation of domain walls (DWs) between domains of opposite polarities where the local magnetization changes its polarity. The DWs can be shifted forward and backward by injecting charge current from left-shift (SL+) and right-shift (SL-) contacts. The new bits are written by first pushing current through shift contacts to move the bits in lockstep fashion to bring the desired bit under write head. Next, spin polarized current is injected through two extra fixed layers in write head (using Write BL and SL) in positive or negative direction to write a ‘1’ or ‘0’ in the NW. Note that conventional write scheme uses write MTJ instead of extra fixed layers. The writing involves current induced spin-torque transfer to flip the magnetization of the free layer (NW in this case). The bits are shifted back to initial state after the write operation. Read is performed by bringing the desired bit under read head using shift and sensing

the resistance of MTJ formed by DW under the read head (using Read BL and BLB in Figure 1).

It should be noted that the resistance of MTJ is high (presented by R_{AP}) when the fixed layer and the free layer are in the antiparallel configuration whereas the resistance is low (presented by R_P) when they are parallel to each other (Figure 1). The bits are shifted back to initial state after the read operation. From the above discussion, it can also be concluded that the read/write involves shifting of bits. For random access, the worst case latency is the summation of number of shifts and read/write latency. However, for serial access the latency is a summation of single shift and read/write latency.

The conventional DWM has high power consumption for write operation. In this work, a shift-based write scheme is used [6] which uses extra fixed layers to shift the bits in the NW instead of nucleating it through MTJ. The resulting write operation is significant low-power and fast at the cost of area overhead (24.7F² per bit vs 2.56F² [6]). The latency, power, and area of DWM are shown in Table 1, where the 1D DWM model [12-13] is used for latency and power estimation.

3. CNN LAYER OVERVIEW

In this section, we first introduce the overview of a simple CNN architecture. Two examples of the embedded memory architectures in CNN convolutional layer, which are cell arrays and input registers, are also discussed.

3.1 CNN Architecture

CNNs are similar to regular neural networks (NNs) since several hidden layers are composed of a set of neurons where each neuron in output layer is connected to the neurons in input layer. While neurons are fully connected in regular NNs, CNNs have local connections by explicitly assuming that the inputs will have spatial locality such as images. The layers of CNNs, unlike regular NNs, have neurons arranged in three dimensions: width, height, and depth. Typical CNNs have five type of layers as shown in Figure 2: INPUT, CONV (convolutional), RELU (rectified-linear units), POOL (pooling), and FC (fully-connected) layer.

3.2 CNN Convolutional Layer Design

The convolutional layer is the core building block of CNNs, which is locally connected to the input volume, and the output volume can be arranged by three hyper-parameters: (a) The depth of the output volume is the number of filters with the same region of the input volume; (b) The stride is the spatial distance between current and next filtering region; and, (c) With sizing zero-padding, the spatial size of the output volume can be controlled,

Figure 2. Conceptual diagram of a simple Convolutional Neural Network (CNN) for CIFAR-10 (32x32x3) classification.

Figure 3. Block diagram of (a) the conventional CNN convolutional layer with (b) Memristor-based Crossbar (MBC) array (c) the conventional input register with line buffer. (d) Update diagram of zero-padding operation of the conventional input register.

Figure 4. Block diagram of 64 dot products with 64 4-bit weights based on (a) MBC [4], (b) SRAM, and (c) DWM. Comparison results of three different implementations in terms of (d) area, (e) power, and (f) energy consumption.

and the spatial size of the input volume can be also preserved. For example, in the CONV2 layer, shown in Figure 2, performs a 5×5 dot product (or filtering) with (output) depth 20, stride ($=S$) 1, and zero-padding ($=P$) 2.

The block diagram of the conventional CNN convolutional layer is shown in Figure 3(a), which is based on the hyper-parameters of CONV2 layer (Figure 2). Figure 3(b) also shows the conventional MBC-based array which consists of Digital-to-Analog Converters (DACs), MBC that performs the weighted summation of the neuron input, and Analog-to-Digital Converters (ADCs). Figure 3(c) shows the conventional input register which is composed of 25 ($=5 \times 5$) DFFs, 4 line buffers with length 11 ($=16-5$), and control logic for zero-padding. A zero-padding operation of the conventional input register in a boundary region is shown in Figure 3(d).

3.3 Multiple Dot Products Implementation

Figure 4(a)-(c) shows the block diagrams of 64 dot product implementation designed with 64 4-bit weights based on MBC, SRAM, and DWM. In Figure 4(a), the MBC-based design con-

sists of MBC array (same as Figure 3(c)) and DAC/ADC array. The SRAM-based design (Figure 4(b)) includes four banks of SRAM and one dot product module which is composed of 64 multipliers and an adder tree. Each SRAM bank stores 64 weights information, where bank index means the bit index of weights. DWM-based design in Figure 4(c) has the similar architecture with SRAM-based design except the memory arrays.

Figure 4(d)-(f) show the comparison results in terms of the area, power, and energy consumption. As shown in the figures, the MBC-based design shows the smallest energy consumption, while the SRAM-based design has the largest area and the largest energy consumptions. In the numerical results shown in Fig. 4, since the throughput of MBC-based design is $64 \times$ higher than others designs, MBC-based design is the smallest and consumes least energy under the same throughput. The DWM-based design consumes more energy (+22%) than MBC-based design, and it has significantly larger area than MBC-based design under the same throughput. We can notice from the numerical results in Figure 4 that the simple replacement of SRAM with DWM does not show any design advantages. *Astute architectural/circuit techniques are*

Figure 5. Block diagram of (a) the proposed DWM-based CNN convolutional layer with (b) 1st accumulator and (c) 2nd accumulator. (d) Update diagram of accumulation operation of the 1st and 2nd accumulators.

Figure 6. Schematic of the proposed DWM-based input register for aligning the inputs of 5×5 (=25) filter.

highly desirable in the DWM based design, which will be presented in the next section.

4. DWM BASED CNN CONVOLUTIONAL LAYER DESIGN

In this section, we present the architectural/circuit-level design techniques for the proposed DWM-based CNN convolutional layer design. By exploiting the unique serial access mechanism of DWM, the input register with sequential access can be replaced with DWM. Additionally, partial implementation of dot product using the resistive cell sensing of DWM achieves low design cost.

4.1 DWM-based CNN convolutional layer

The architecture of the proposed DWM-based CNN convolutional layer is illustrated in Figure 5(a). It consists of DWM-based input registers, DWM-based cell arrays, adder modules, the 1st/2nd accumulators, and Add bias modules. The detailed architecture of the 1st and 2nd accumulators are shown in Figure 5(b) and (c), respectively. DWM-based CNN convolutional layer has similar architecture as the conventional one in Figure 3(a) except the bit-serial architecture based on two additional accumulators, which are located between adder module and Add bias module and accumulate in every bit-serial operation.

4.2 DWM-based input register

The conventional input register generally supports the simultaneous read and write operations as shown in Figure 3(c). The shift-based access pattern of DWM is efficiently exploited to implement the input registers at low cost. Figure 6 shows the proposed DWM-based input registers. Since the DWM cannot perform the simultaneous write and read operations, the input register is divided into two NWs, and each NW has one write head and 25 read heads to implement 5×5 filtering operation. Since the bit-serial architecture is employed, the length of each NW is 4 (=half bit width of input) times of the total length of the conventional input register in a 1D form. The control for zero-padding of DWM-based input register works in the same way as the conventional input register.

4.3 DWM-based cell array

In the DWM based architecture shown in Figure 4(b), DWM array and dot product operators are separately implemented. In the proposed architecture (Figure 7), partial dot products are merged together in the cell arrays where all CNN weights are loaded. The DWM-based cell array consists of DWM cell sub-arrays and ADC sub-arrays, and the current reference circuit and/or several adders are located outside the cell array. The detailed operations of the DWM-based cell array are explained in the following sub-sections.

4.3.1 DWM-based cell string

The DWM-based cell array architecture shown in Figure 7 implement the dot product operation, where the number of weights is 25 with the output depth of 20. The operation can be considered as 20 parallel filtering operations with each filter having 25 filter taps with shared inputs (bit width of each weight=16 and input bit width =8). The operation can be expressed as

$$y_l[m][n] = \sum_{k=0}^{24} x_k[m] \cdot w_{k,l}[n]. \quad (1)$$

, where l = output depth index, k = weight index, m = the bit index of input, n = the bit index of weight, and w = weight. In Figure 7, the blue box shows an example of $\sum_{k=0}^6 x_k[m] \cdot w_{k,0}[n]$ with bit-serial operations. The operation in blue box is $x_0[m] \cdot w_0[n] + x_1[m] \cdot w_1[n] + x_2[m] \cdot w_2[n] + x_3[m] \cdot w_3[n] + x_4[m] \cdot w_4[n] + x_5[m] \cdot w_5[n] + x_6[m] \cdot w_6[n]$.

Figure 7. Schematic of the proposed DWM-based cell array (the number of weights = 25, the bit width of weights = 16, the depth of output = 20, and the length of DWM cell string = 7).

Figure 8. Schematic of (a) voltage reference and 3-bit ADC with (b) voltage comparator, (c) *diff_lat*, and (d) *diff_lat2*. (e) Kickback noise simulation. (f) The limitation of length of DWM-based cell string due to large stacking effect (HSPICE simulation).

$w_4[n] + x_5[m] \cdot w_5[n] + x_6[m] \cdot w_6[n]$, where m changes from 0 to 7 and n changes from 0 to 15 in bit-serial manner. As shown in the figure, each of $w_k[n]$ is stored in MTJ and $x_k[m]$ is applied to the gate input of the selective transistor. The output of $\sum_{k=0}^6 x_k[m] \cdot w_k[n]$ operation can be modeled as series of resistors (referred as the DWM based cell string in the figure), where the resistance values are dependent on the input $x_k[m]$ and weight $w_k[n]$.

As shown in Figure 7, the DWM-based cell string is composed of serial connection of selective transistor and MTJ pairs. Since MTJ cell can be modeled as R_P ('0' in logic) or R_{AP} ('1' in logic) resistor, and the selective transistor has R_{on} or open state depending on the gate input $x_k[m]$, each pair can be a resistor with four resistance value – R_P , R_{AP} , ($R_{on} \parallel R_P$), and ($R_{on} \parallel R_{AP}$). By assuming that $R_{on} < R_P < R_{AP}$ and ($R_{on} \parallel R_P$) $\approx$ ($R_{on} \parallel R_{AP}$) in this work, each pair can be modeled as a resistor with three variable resistance, $R_{AP} > R_P > R_S$ where $R_S = R_{on} \parallel R_P$ (or R_{AP}). Here, the selective transistor performs a masking operation for the MTJ value based on the gate input. The logic value of the pair is same as AND gate output between the inverted input and the bit stored in MTJ (=the bit information of weights). The DWM-based cell string accumulates the resistance values of these pairs, and the accumulated resistance value is same as the number of '1s' in terms of logical value. As a result, DWM-based cell string for 25 weights performs (1). The output of DWM-based CNN convolutional layer with input depth of $p=16$, can be expressed as the following equation:

$$A_l[m][n] = \sum_{p=0}^{15} y_{l,p}[m][n]. \quad (2)$$

$$z_l = \sum_{m=0}^7 \left(-A_l[m][0] + \sum_{n=1}^{15} A_l[m][n] \cdot 2^{-n} \right) \cdot 2^m + b_l. \quad (3)$$

, where A = the output of adder module in Figure 5(a), b = bias, and z = the output of DWM-based CNN convolutional layer.

As shown in Figure 7, DWM-based cell array reads out weights $w_k[n]$ through shift operation. With the bit-serial operations, LSB-first and MSB-first sequence are alternatively used. As presented in in Figure 5(b), the 1st accumulator operates in LSB-first mode or MSB-first mode depending on MODE select input, and those operation are illustrated in Figure 5(d).

4.3.2 Voltage Reference Circuit

As presented in the previous subsection, the resistance of the DWM-based cell string may vary depending on the input $x_k[m]$ and weight $w_k[n]$. The varying resistance value will appear as the changes of the voltages on 'read BL' in Figure 7. Therefore, we need to generate the reference signals for sensing read BL voltages. The following equation presents the reference resistance $R_{REF(i)}$ for sensing read BL:

$$R_{REF(i)} = (i + 0.5)R_{AP} + (N - (i + 0.5) - j)R_P + jR_S \quad (4)$$

$$= (i + 0.5)\Delta + \{(N - j)R_P + jR_S\},$$

, where i = reference index, j = the number of selective transistors turned on, N = the length of DWM-based cell string, $R_S = R_{on} \parallel R_P$ (or R_{AP}), and $\Delta = R_{AP} - R_P$. As presented in Figure 8 (a), the reference voltage REF_i can be calculated by multiplying the constant reference current I_{REF} and reference resistance $R_{REF(i)}$, and if the number of '1s' on read BL is i , $R_{BL(i)}$ represented in the resistance value, which satisfy the following inequality:

$$\begin{aligned} R_{BL(0)} &< R_{REF(0)} \\ R_{REF(i)} &< R_{BL(i+1)} < R_{REF(i+1)} \\ R_{REF(N-1)} &< R_{BL(N)} \end{aligned} \quad (5)$$

, where $i = 0 \dots N-2$. According to (4), the voltage reference circuit in the left side of Figure 8(a) is composed of input-independent resistor ladder and input-dependent DWM-based cell string consisting only of MTJ cells in the parallel state (R_P represented in the resistance value), which can compensate the jR_S term.

4.3.3 3-bit ADC based on flash type

Figure 8(a) shows 3-bit flash type ADC with the voltage reference circuit, where typical flash ADC can be divided into three stages: (a) in first step, thermometer code is generated using comparators; (b) thermometer code is converted to one-hot code using the gates logic (e.g. AND-NOT gate); and, in the third step, (c) the outputs of the gates are decoded as the binary output. To generate the thermometer code, we used the voltage comparator with low kickback noise in Figure 8(b) [14]. This is to ensure voltage margin as shown in Figure 8(e). For the thermometer-to-one hot conversion, the modified CMOS differential latch (*diff_lat2*) in Figure 8(d) is used that is made with only four additional transistors based on differential latch (*diff_lat* or N-C²MOS latch [15]) shown in Figure 8(c). Finally, the conventional ROM-based design is employed to obtain binary outputs from one-hot code.

Figure 9. Comparison results between the conventional MBC-based and the proposed DWM-based CNN convolutional layer implementations in terms of (a) area and (b) energy consumption under same throughput condition (bit width=4-bit).

4.3.4 Limitation of DWM-based cell string

As shown in Figure 8(f), the difference of read BL voltages between case 1 and case 2 is larger than the reference voltages difference between case 1 and 2. Since DWM-based cell string in the voltage reference circuit has only R_P of MTJ cells (however, the DWM-based cell sub-array has both R_P and R_{AP}), the reference voltage cannot completely track the read BL of MTJ cells. This effect is amplified with long length of DWM-based cell string. Considering our simulation results and the bit width of binary output, the length of DWM-based cell string is limited to 7. When the number of weights is more than 8, additional adders between the outputs of ADCs are required, however, this overhead is minor considering the overall design cost (in local-connected CNNs).

5. NUMERICAL RESULTS

DWM-based cell array has been designed and circuit simulations are performed using 65 nm CMOS technology and standard cell library. Using the liberty file (.lib) format, we generate a DWM cell library through Synopsys for delay, power and area estimation. The delay of DWM has been calculated based on bitcell delay, interconnect resistances and capacitance estimated for 65nm process. The large DWM (shift-based write) footprint is 24.75 F²/bit.

For the comparisons of power dissipation in architectural/circuit-level, PrimeTime-PX and HSPICE are used in the simulations with TYPICAL 1.2V 25°C corner @ 333 MHz (clock period=3ns).

Figure 9(a)-(b) show the comparison between the conventional MBC-based and the proposed DWM-based convolutional layer implementations in terms of the area and energy consumption. The numerical results of the MBC-based design is obtained from the previous work [4]. In our implementation, the depth of the input layer is 16, the depth of the output layer is 20, and the bit-width of input and weight are 4 bit, which is same with the previous work [4] for comparison. As shown in Figure 10, the DWM-based design achieves 34% of energy savings under same throughput condition. The energy savings is mainly due to the low cost DWM-based implementation of input registers and the reduced DAC cost. Under same throughput condition, the proposed DWM-based design shows around 2× larger area than the MBC-based design, which is mainly due to the large digital adder tree circuits. Although the area is larger, the bit-width of the proposed DWM-based design is easily extendable, which is adequate for high accuracy deep learning applications. One of the largest advantages of the proposed DWM based design over the conventional memristor based design is that the bit-width of the images (input and layers) and parameters (weight and bias) are scalable. Figure 10(a) shows the accuracy loss of the proposed design with

Figure 10. Various fixed-point results of the proposed DWM-based CNN convolutional layer implementations in terms of (a) accuracy loss and (b) energy consumption.

scalable bit-width. As presented in Figure 10(a), the accuracy loss means a testing accuracy degradation of fixed-point simulations compared to floating-point simulations, and 2,751 of CIFAR-10 test images are used for CIFAR-10 image classification. While the conventional memristor based approach can be designed only with the fixed bit-width of 4-bit (bit-width is not scalable) having the accuracy loss of 31.6%, the proposed design can achieve accuracy loss up to 0.8% with bit-width scalability. Figure 10(b) also shows the increasing energy consumptions with the scalable bit-width.

6. CONCLUSIONS

Bit-width extendability, energy, and performance of a convolutional layer of CNNs are crucial for future deep learning accelerators. We investigated the DWM-based design of input registers and the cell array in CNN convolutional layer, and coupled their requirements such as serial access and low-cost dot product architecture with the corresponding features of emerging DWM for bit-width extendability and energy-efficiency. The proposed designs reveal large energy savings compared to the conventional MBC-based design approach under the same throughput.

7. REFERENCES

- [1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [2] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [3] S. G. Ramasubramanian *et al.*, "SPINDLE: SPINtronic deep learning engine for large-scale neuromorphic computing," in *Proc. ISLPED*, 2014, pp. 15–20.
- [4] X. Liu *et al.*, "RENO: A high-efficient reconfigurable neuromorphic computing accelerator design," in *Proc. DAC*, 2015, pp. 1–6.
- [5] L. Cavigelli *et al.*, "Origami: A Convolutional Network Accelerator," in *Proc. GLSVLSI*, 2015, pp. 199–204.
- [6] R. Venkatesan *et al.*, "DWM-TAPESTRI - An energy efficient all-spin cache using domain wall shift based writes," in *Proc. DATE*, 2013, pp. 1825–1830.
- [7] F. Alibart *et al.*, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.
- [8] S. Ghosh, "Path to a TeraByte of on-chip memory for petabit per second bandwidth with < 5Watts of power," in *Proc. DAC*, 2013, pp. 1–2.
- [9] R. Venkatesan *et al.*, "TapeCache: a high density, energy efficient cache based on domain wall memory," in *Proc. ISLPED*, 2012, pp. 185–190.
- [10] H. Yu *et al.*, "Energy efficient in-memory machine learning for data intensive image-processing by non-volatile domain-wall memory," in *Proc. ASP-DAC*, 2014, pp. 191–196.
- [11] C. Augustine, "Spintronic memory and logic: From atoms to systems," Ph.D. dissertation, Elect. Eng., Purdue Univ., West Lafayette, IN, 2011.
- [12] M. Hayashi, "Current driven dynamics of magnetic domain walls in permalloy nanowires," Ph.D. dissertation, Mater. Sci. Eng., Stanford Univ., Stanford, CA, 2006.
- [13] A. Iyengar and S. Ghosh, "Modeling and analysis of domain wall dynamics for robust and low-power embedded memory," in *Proc. DAC*, 2014, pp. 1–6.
- [14] R. Jacob Baker, *CMOS Circuit Design, Layout, and Simulation*, 3 ed. Hoboken, NJ, USA: Wiley, 2010.
- [15] J. Kim *et al.*, "CMOS sense amplifier-based flip-flop with two N-C2MOS output latches," *Electron. Lett.*, vol. 36, no. 6, pp. 498–500, Mar. 2000.