EDA Challenges for Memristor-Crossbar based Neuromorphic Computing

Beiye Liu, Wei Wen, and Yiran Chen
ECE Department
University of Pittsburgh
Phone: (412) 624-5836
{bel34, wew57, yic52}@pitt.edu

Xin Li
ECE Department
Carnegie Mellon University
Phone: (412) 268-6507
xinli@cmu.edu

Chi-Ruo Wu and Tsung-Yi Ho
CSIE Department
National Cheng Kung University
Phone: +886-6-275-7575
{gtrw, tyho}@csie.ncku.edu.tw

ABSTRACT

The increasing gap between the high data processing capability of modern computing systems and the limited memory bandwidth motivated the recent significant research on *neuromorphic computing systems* (NCS), which are inspired from the working mechanism of human brains. Discovery of memristor further accelerates engineering realization of NCS by leveraging the similarity between synaptic connections in neural networks and programming weight of the memristor. However, to achieve a stable large-scale NCS for practical applications, many essential EDA design challenges still need to be overcome especially the state-of-the-art memristor crossbar structure is adopted. In this paper, we summarize some of our recent published works about enhancing the design robustness and efficiency of memristor crossbar based NCS. The experiments show that the impacts of noises generated by process variations and the IR-drop over the crossbar can be effectively suppressed by our noise-eliminating training method and IR-drop compensation technique. Moreover, our network clustering techniques can alleviate the challenges of limited crossbar scale and routing congestion in NCS implementations.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles–*VLSI (very large scale integration).*

Keywords

Memristor Crossbar; Neuromorphic Computing; Neural Networks;

1. INTRODUCTION

As Moore's Law is approaching its end [14], technology scaling of conventional CMOS devices slows down and many studies on emerging circuits and devices are being performed. Two promising emerging nano-devices, i.e., spintronic [15] and resistive devices (a.k.a. memristor) [16], draw considerable attentions from the researchers. Besides, the gap between CPU computing capacity and memory accessing bandwidth greatly limits the up-scaling of von Neumann computer architecture, known as "memory wall" problem [17].

Emerging memristor devices has been well received as a promising candidate for next-generation memory [16]. Crossbar structure of memristors can be manufactured with a high integration density at a level of 10^{10} synapses per square inch; the achieved power consumption is also as low as one trillion operations per second (TOPS) per Watt [8][5]. Moreover, the non-volatility and programmability of memristor resistance can efficiently mimic the variable synaptic strengths of biological synapses, using memristor to implement artificial neural network has become a hot research topic [7].

The VLSI realization of brain-inspired computing systems is referred to as neuromorphic computing systems (NCS). Although memristor has demonstrated many attractive characteristics in NCS implementation, there are still many fundamental challenges in designing a robust and efficient NCS. For instance, the computing and training reliability of a memristor crossbar is often deteriorated by process variations and signal noises [5]. As another example, voltage distributed over a crossbar degrades due to wire resistance (a.k.a. IR-drop), which could become severer as the crossbar size increases and therefore, constrain the scalability of the crossbar structure. Finally, a practical neural network is usually so large that one single crossbar may not offer sufficient connections. Utilizing multiple crossbars will raise the routing congestion problem; on the contrary, using memristor to implement a sparse neural network may cause a low crossbar utilization rate.

In this paper, we summarize some of our recent published works about enhancing the design robustness and efficiency of memristor crossbar based NCS.

2. PRELIMINARY

2.1 Memristor Based NCS

Figure 1 illustrates the structure of a feed-forward neural network where input (x_i) neurons are connected with output neurons (y_i) through synapses (w_{ij}). The output message at every output neuron is activated after collecting and weighting messages from all the inputs. In a memristor-based NCS, every synapse is implemented by a memristor and the synaptic weight is represented by the resistance (memristance) of the memristor device. Mathematically, the computation can be formulated as:

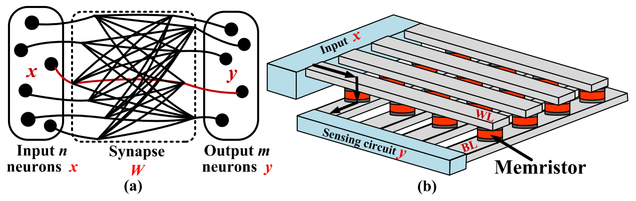


Figure 1. (a) Conceptual overview of a neural network [8]. (b) Circuit architecture of a memristor crossbar [7].

$$y_n = W_{n\times m} \times x_m, \tag{1}$$

where x_m and y_n are the activity patterns of m input neurons and n output neurons, respectively, and $W_{n\times m}$ is the weight matrix denoting synaptic strengths between inputs and outputs. Matrix-vector multiplication in Eq. (1) is the core operation of one of the commonest computation (namely, "recall" process) in NCS. Both network of discrete memristors and memristor crossbar can realize this operation but crossbar is considered as a more efficient way because of its structural similarity [6]. As Figure 1(b) shows, in a recall process, x is mimicked by the input voltage vector applied on the word-lines (WLs), and the bit-lines (BLs) are grounded. Each memristor is programmed to a resistance state representing the weight of the correspondent synapse. The current along each BL is collected and converted to the output voltage vector y by the "neurons" at the output of the crossbar [5].

2.2 Training of Memristor Crossbar

"Training" is another basic computation of NCS. The training of a NCS can be categorized as either close-loop on-device (OLD) training or open-loop off-device (OLD) training. In OLD training, amplitudes and durations of the programming signals are first computed based on different algorithms and applications. Then the memristors in the crossbar are programmed accordingly. On the other hand, CLD is an iterative scheme to update weights (resistances) of memristors according to the feedback from outputs [10].

When programming a memristor crossbar, programming pulses with different amplitudes and durations are applied to the crossbar for a desired resistance matrix $\boldsymbol{R}$: the voltage of the WL and BL connecting the target memristor ($\boldsymbol{Rij}$) are set to $+V_{bias}$ and GND, respectively, while all other WLs and BLs are connected to $+V_{bias}/2$. Hence, only $\boldsymbol{Rij}$ is applied with a full V_{bias} above the threshold that can change the device's resistance state, leaving the rest of memristors remain unchanged because they are only "half-selected" with a voltage of $V_{bias}/2$ [11].

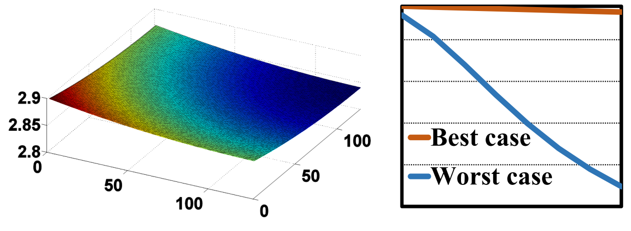
3. Design Challenges Overview

In this section, we discuss some challenges in the design and design automation of memristor-based NCS.

3.1 IR-drop

The resistance along the metal wires connected to the memristors in the crossbar causes the IR-drop issue in NCS: the voltage reaching the device could be considerably lower than that supplied by the write driver. In general, the influence of IR-drop on a memristor is determined by its position in the crossbar as well as the resistance states of all memristors. As proved in [11], when all memristors are at their low-resistance state (LRS), both recall and training processes of the memristor crossbar will encounter severe reliability problem, especially when the size of the crossbar is beyond 64×64.

Figure 2 virtualizes the IR-drop issue on a crossbar with a size of 128×128. Figure 2(a) depicts the programming voltage distribution ($\boldsymbol{V'}$) in the training process. The programming voltage from the driver is *2.9V*. $\boldsymbol{V'}_{ij}$ is the voltage reaching the memristor between WL$_i$ and BL$_j$. The largest IR-drop happens at the far end location as $\boldsymbol{V'}_{(128,128)}$. The worst case and the best case of IR-drop occur when all the memristors are at their HRS (high-resistance state) and LRS, respectively, as shown in Figure 2(b). IR-drop greatly harms the programmability of the memristor crossbar and degrades the computation accuracy of the NCS. Similar trend can be also found during recall process of the NCS.



(a) (b)

Figure 2. (a) Programming voltage distribution on a 128×128 all-HRS-memristor crossbar (the best case). (b) Programming voltage degradation vs. crossbar size [5].

IR-drop significantly affects reliability of the recall and training processes of NCS. During the recall process, input patterns (voltages) are applied on the WLs and outputs are read from the BLs. Because of IR-drop, the BLs that are far from the input driver are subject to the distortion of the voltage applied on the memristors on the BLs and therefore, the distortion of the outputs. During the training process, resistance matrix $\boldsymbol{R}$ of all memristors should be programmed to the value representing the targeted weight matrix. For a memristor initially at HRS state, its final resistance $\boldsymbol{R}_{ij}$ is a function of the voltage ($\boldsymbol{V}_{ij}$) and the accumulative duration ($\boldsymbol{T}_{ij}$) of the programming pulse such as:

$$\boldsymbol{R}_{ij} = f(\boldsymbol{T}_{ij}, \boldsymbol{V}_{ij}, \boldsymbol{R}_{HRS}). \tag{2}$$

Due to IR-drop, the actual voltage applied on the memristor is $\boldsymbol{V'}_{ij}$ that is different from the targeted $\boldsymbol{V}_{ij}$ and thus, leading to an inaccurate training result (weight matrix).

3.2 Routing Challenges of NCS Design

In contrast to the limited scale of memristor crossbars that can be manufactured presently, neural networks used in real applications are often very large. For example, DNN (deep neural network) used in [1] is composed of more than 4000 input neurons. The network designed for LDPC coding in IEEE 802.11 also has a similar scale [2]. To implement such large-scale neural networks, three memristor-based schemes can be utilized: (1) pure *discrete synapse (memristor) design* (DSD), (2) pure *multiple crossbar design* (MCD) and (3) a hybrid design trading off between (1) and (2). The DSD is flexible to realize synaptic connections but its area cost increases exponentially with the neuron number because of routing congestion. Inter-crossbar routing congestion is also a major problem in MCD. Moreover, if the neural network is sparse, implementing the network with MCD may cause a low utilization of (the synaptic connections of) the memristor crossbar. We note that the routing congestion and low crossbar utilization problems are very common in some applications with large and sparse neural networks. For example, in LDPC coding based on back propagation algorithm, less than 1% possible connections in the network are actually connected [2]. Considering the pros and cons of DSD and MCD, A hybrid scheme (DSD+MCD) can be promising to overcome these challenges. Given a large and sparse neural network, we proposed a clustering technology to concentrate randomly distributed connections into multiple small clusters. As a result, within-cluster connections are dense and between-cluster

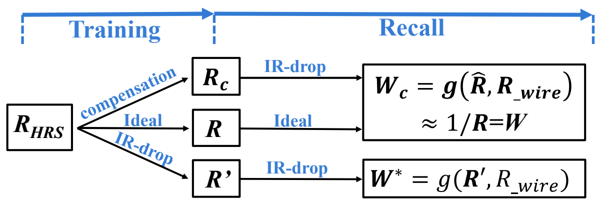


Figure 3. Compensation for both training and recall processes [5].

ones are sparse. MCD and DSD respectively realize within- and between-cluster connections, resulting in both high crossbar utilization and low routing congestion. More details will be given in the next section.

4. Solutions for NCS Design Challenges

We proposed some techniques to solve the design challenges of NCS illustrated in Section 3.

4.1 IR-drop Compensation

To alleviate IR-drop issue, compensation techniques can be utilized in recall and training processes of memristor crossbar. Figure 3 gives an overview of IR-drop compensation scheme in [5].

Taking into account the impact of IR-drop during the recall, we define $\boldsymbol{W^*}$ as the actual but distorted weight matrix. $\boldsymbol{W^*}$ is determined by the targeted memristor resistance state $\boldsymbol{R}$ and the wire resistance $R_{\_wire}$ as:

$$\boldsymbol{W}^* = g(\boldsymbol{R}, R_{\_wire}). \quad (3)$$

As shown in Figure 3, the IR-drop compensation scheme optimizes a compensating resistance state $\boldsymbol{R_c}$ that generates a weight matrix $\boldsymbol{W_c}$ close to the ideal target $\boldsymbol{W}$ as:

$$\min_{\boldsymbol{R}_c}\|\boldsymbol{W}-\boldsymbol{W}_c\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}(\boldsymbol{W}_{(i,j)} - \boldsymbol{W}_{c_{(i,j)}})^2. \quad (4)$$

Gradient descent algorithm can optimize this target as:

$$\boldsymbol{R}_{c\_k+1} = \boldsymbol{R}_{c\_k} - \gamma\sum_{i=1}^{n}\sum_{j=1}^{m}(2\left(\boldsymbol{W}_{(i,j)} - \boldsymbol{W}_{c_{(i,j)}}\right)\frac{\partial \boldsymbol{W}_{c(i,j)}}{\partial \boldsymbol{R}_{c\_k}}) \quad (5)$$

Here $\boldsymbol{W}_c = g(\boldsymbol{R_c}, R_{\_wire})$ and can be explicitly measured [5].

Different from the recall process, the optimization target of IR-drop compensation in the training process is minimizing the difference between the trained $\boldsymbol{R'}$ and the targeted $\boldsymbol{R}$ that represents desired weight matrix $\boldsymbol{W}$. As aforementioned, all memristors can be set towards HRS to minimize the IR-drop impact during the training process. As Eq. (2) formulated, the final resistance of a memristor is the function of the programming pulse voltage and duration. Then the IR-drop can be compensated by prolonging the programming duration.

4.2 Memristor Crossbar Reduction

Another method to alleviate the impact of IR-drop is to reduce the scale of the involved computation and thus, the required size of the memristor crossbar [5]. The proposed crossbar reduction scheme is to approximate the weight matrix $\boldsymbol{W}$ $(n\times m)$ in Eq. (1) by leveraging singular value decomposition (SVD) as [4]:

$$\boldsymbol{W} = \boldsymbol{U}\sum\boldsymbol{V} \approx \boldsymbol{W}_{appx} = \sum_{i=1}^{r}\delta_i\cdot\boldsymbol{u}_i\cdot\boldsymbol{v}_i. \quad (6)$$

Here $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices, $\boldsymbol{\Sigma}$ is an rectangular diagonal matrix with singular values of $\boldsymbol{W}$. δ_i (i=1,…r) are the first r (i.e., the rank of $\boldsymbol{W}_{appx}$) singular values of $\boldsymbol{W}$. $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the approximated left and right singular vectors of $\boldsymbol{W}$, respectively. The difference between $\boldsymbol{W}$ and $\boldsymbol{W}_{appx}$, that is $\Delta\boldsymbol{W} = \|\boldsymbol{W}-\boldsymbol{W}_{appx}\|$, is decided by the coverage of $\sum_{i=1}^{r}\delta_i$ on the overall summed $\sum_{i=1}^{m}\delta_i$. The difference, hence, $\Delta\boldsymbol{W}$ can be controlled by the value of r. [5] gives the strategy to select r.

Based on the approximation result, we are able to transform the weight connection function in Eq. (1) to:

$$\begin{aligned}\boldsymbol{W}\cdot\boldsymbol{x} &\approx (\textstyle\sum_{i=1}^{r}\delta_i\cdot\boldsymbol{u}_i\cdot\boldsymbol{v}_i)\cdot\boldsymbol{x}\\ &= \boldsymbol{W_{left}}\cdot\boldsymbol{W_{right}}\cdot\boldsymbol{x}\end{aligned} \quad (7)$$

where

$$\boldsymbol{W}_{left} = [\delta_1\cdot\boldsymbol{u}_1 \ldots \delta_r\cdot\boldsymbol{u}_r],\ \boldsymbol{W}_{right} = \begin{bmatrix}\boldsymbol{v}_1\\ \vdots\\ \boldsymbol{v}_r\end{bmatrix} \quad (8)$$

Here $\boldsymbol{W}$ was originally represented on an $n\times m$ crossbar and $m\times 1$ vector $\boldsymbol{x}$ is represented by the input voltage vector. Eq. (7) and (8) show that the connection function can be transformed to a new two-stage system that consists of a $n\times r$ weight matrix $\boldsymbol{W}_{left}$ and a $r\times m$ weight matrix $\boldsymbol{W}_{right}$. Note that $r \ll n$ or m. This method is named as one-dimensional (1-D) reduction.

Figure 4 illustrates the programming voltage distribution on a 128×128 memristor crossbar when IR-drop is considered. Here all memristors are at LRS to demonstrate the worst-case impact of IR-drop. We highlighted (colored) the memristor locations with a voltage drop higher than V_{bias} /2 under "half-selected" programming scheme introduced in Section 2.2. We name the boundary of the highlighted area as the "hard-limit". Any memristor outside the "hard-limit" will not be effectively programmed because they are practically "half-selected". Increasing the programming voltage to raise the voltage applied on the memristors outside the "hard-limit", however, will affect the memristors that should be "half-selected". Hence, the scale of the "hard-limit" serves as a good measurement of programming robustness of the memristor crossbar. As shown in Figure 4(a), the size of the largest "hard-limit" is 48×48, or say, the maximum dimension of the data that can be processed is only 48. Our "2-D reduction method" proposed in [5] can reduce the required memristor crossbar size at both dimensions. However, by reducing the size of one dimension down to a smaller value, say, r = 22, the size of another dimension can be extended to 128, as depicted in Figure 4(b). Such a crossbar is sufficient to process the data with a size of 128 leveraging our proposed 1-D reduction method, as long as the rank of $\boldsymbol{W}_{appx}$ is not higher than 22.

4.3 Neural Network Partitioning

Spectral graph partitioning (also spectral clustering) is a candidate to partition the graph which representing a neural network into multiple groups. The optimization target is to minimize the between-group similarity and maximize the within-group similarities [3]. As the message propagation direction on the neural network has no impact on the optimization function, the neural network can be abstracted as an undirected graph to perform spectral partitioning. In NCS design, by specifying the "similarity" as "the number of connections", the goal of our spectral graph partition-

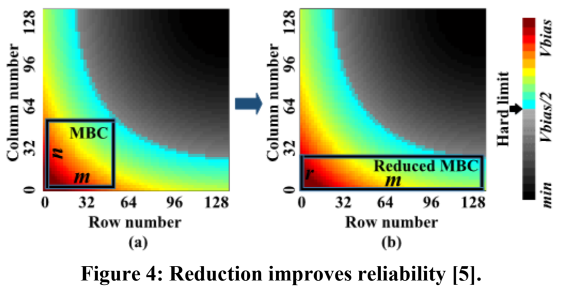


Figure 4: Reduction improves reliability [5].

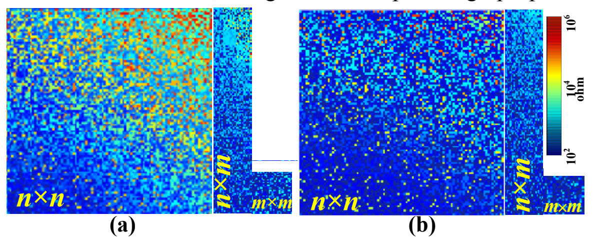


Figure 5. Trained memristor resistance discrepancy (a) without IR-drop compensation. (b) with IR-drop compensation [6].

ing becomes minimizing the (between-group) connections that need to be mapped to DSD and maximizing the (within-groups) connections that fit into the crossbars (MCD), thus, decreasing the routing congestion and increasing the crossbar utilization.

5. Experiments

5.1 Evaluations of IR-drop Compensation and Crossbar Reduction

Figure 5 shows our experiment results about the resistance discrepancy between the targeted crossbar and the actual trained one impacted by IR-drop and process variations. In our experiment, the programmed memristor resistance is assumed to follow the log-normal distribution as $r = r_0 \cdot \exp(\theta)$ [12], where, $\theta \sim N$ (0, σ) is Gaussian distribution and r_0 is the mean value.

We use Hopfield network as an example to illustrate the effectiveness of IR-drop compensation and crossbar reduction schemes. A crossbar with a scale of 128×128 (original $n \times n$) can be reduced to 128×19 ($n \times r$) by applying 1-D reduction and to 19×19 ($r \times r$) by 2-D reduction scheme [5]. Here r is set to 15% of n, or the maximum pattern numbers that can be stored in a 128×128 Hopfield network in theory [13]. As Figure 5(a) shows, reduction schemes significantly reduce the resistance discrepancy as the crossbar size decreases, resulting in a higher quality of training. Moreover, as shown in Figure 5(b), the IR-Drop compensation scheme effectively minimizes the crossbar resistance discrepancy. More experiments also showed that the training quality enhancement could substantially improve recall successful rate of the NCS [5].

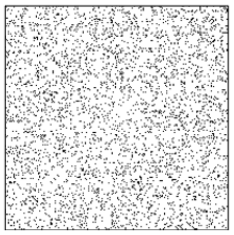
5.2 Evaluations of clustering

Figure 6(a) and (b) demonstrates the adjacency matrix of a Hopfield network with 200 neurons before and after spectral partitioning, respectively. In the figure, the i-th neuron is indexed on the i-th row (also on the i-th column). A black element at location (i, j) shows the connection between the i-th and the j-th neurons, and an empty (white) space shows no connections. The connection topologies in (a) and (b) are the same and the only difference is the order of the neurons. Reordering neurons through spectral partitioning, clusters are formed in the red squares. Experiment shows that the connections are efficiently concentrated into several dense clusters, which can be efficiently implemented by MCD.

In a hybrid design (DSD+MCD), however, there are still some fundamental challenges need to be solved: Firstly, the size of a formed cluster should not exceed the maximum size of the available crossbars. Conventional spectral partitioning [3], however, does not consider this limitation; Secondly, it is usually difficult to cluster majority of the connections into clusters by simply performing spectral partitioning algorithm because random neural networks may not have a good clustering property. For instance, 52% of total connections in Figure 6(b) are still outside clusters.

6. Conclusion

In this paper, we summarized some EDA challenges in neuromorphic computing system design, such as the IR-drop issue and the routing congestion and low crossbar utilization of large-scale sparse neural network implementation. We also introduced some techniques, i.e., IR-drop compensation, crossbar reduction and neural network clustering to overcome these challenges.


(a) Original

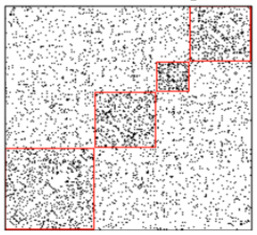

(b) Clustered

Figure 6. Connection matrices before and after partitioning.

7. Acknowledgement

This work is partially supported by AFRL FA8750-14-1-0241, NSF CNS-1116171, and HP Lab Innovation Research Program.

8. REFERENCES

[1] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *CVPR*, pp. 3642-3649, 2012.

[2] "IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11*-2012, pp. 1-2793 , 2012.

[3] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395-416, 2007.

[4] Golub, G. H. and Reinsch, C., "Singular value decomposition and least squares solutions," *Numerische Mathematik*, 1970.

[5] B. Liu, *et al.*, "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," *ICCAD*, pp. 63-70, 2014.

[6] B. Liu, *et al.*, "Digital-assisted noise-eliminating training for memristor crossbar-based analog neuromorphic computing engine," *DAC,* pp. 1-6, 2013.

[7] M. Sharad, D. Fan, and K. Roy, "Ultra low power associative computing with spin neurons and resistive crossbar memory," DAC, pp. 1-6, 2013.

[8] M. Hu, *et al*, "Hardware realization of BSB recall function using memristor crossbar arrays," DAC, pp. 498-503, 2012.

[9] P. F. López, *et al*, "A Computational Study of the Diffuse Neighbourhoods in Biological and Artificial Neural Networks," *IJCCI*, pp. 490-495, 2009.

[10] Miao Hu, *et al*, "BSB training scheme implementation on memristor-based circuit," *CISDA*, 2013.

[11] J. Liang and H. S. Wong, "Cross-point memory array without cell selectors—device characteristics and data storage pattern dependencies," *, IEEE Transactions on Electron Devices,* vol. 57, pp. 2531-2538, 2010.

[12] S. R. Lee*, et al.*, "Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory," *VLSIT*, pp. 71-72, 2012.

[13] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences of the USA*, 1982.

[14] A. Asenov, S. Kaya, and A. R. Brown, "Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness," *Electron Devices, IEEE Transactions on,* vol. 50, pp. 1254-1260, 2003.

[15] X. Dong, *et al*, , "Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement," *DAC* , pp. 554-559, 2008.

[16] D. Niu, Y. Chen, C. Xu, and Y. Xie, "Impact of process variations on emerging memristor," DAC, pp. 877-882, 2010.

[17] S. A. McKee, "Reflections on the memory wall," *the 1st conference on Computing frontiers*, p. 162, 2004.