2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)

Design of an optimized CMOS ELM accelerator

Manoj Kumar Sharma, Umesh Chandra Lohani, Vivek Parmar and Manan Suri*
Department of Electrical Engineering, Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India-110016
*manansuri@ee.iitd.ac.in

***Abstract*—In the last decade, artificial intelligence (AI) has emerged at the forefront of driving many technological innovations. A variety of algorithms have been proposed as possible alternatives to implement AI in computing systems. Extreme learning machine (ELM) has emerged as one of the most effective training algorithms for simple applications based on single layer feed-forward neural networks (SLFN) because of its unique training method. Hardware implementation of neural network algorithms is a critical requirement for deploying them in time-sensitive applications. In this paper, we present a simplified AI accelerator based on CMOS technology that implements an ELM based inference engine. We present analysis of implementing such an accelerator on different technology nodes with a comparative analysis to analyze the impact of technology node scaling on performance of the proposed accelerator in terms of power and area. For the analysis, the workload used was a network of dimensions 81x18x1. We observed a remarkable benefit in speed (1.3x), area (14x) and power (7x) by scaling the design from 180 nm to 45 nm. Further, we present an analysis showing benefits of introducing emerging non-volatile memory (NVM) technologies like RRAM as the primary memory technology for the accelerator. The analysis shows that replacing the conventional CMOS with RRAM would give significant benefits in leakage (4.5x) and area (33x).**

***Index Terms*—AI, Neural networks, Accelerator, Neuromorphic computing, ELM, NVM**

I. Introduction

In the last decade there has been growing interest in the hardware implementation of intelligent systems. Feed forward neural networks are being explored for implementation of these systems due to their special characteristics like fast learning, better characterization and classification capabilities [1]. These technologies are very effective for various applications like pattern classification, clustering / characterization, function approximation, prediction / forecasting etc. The main limitation in deploying these algorithms is their computational complexity. Commonly used machine learning algorithms such as SVM (support vector machines), MLP(multilayer perceptrons) etc. require a significant amount of computational power and memory. Due to this, it becomes difficult to deploy on systems with low power requirements such as IoT (Internet of Things) devices which use a micro-controller as the main computational unit. Also the memory and computational power limit the ability to map these algorithms completely to hardware. A new feed-forward network algorithm called extreme learning machine (ELM) can overcome these limitations associated with the more traditional neural network algorithms due to its simplistic single hidden layer based implementation and its ability to model arbitrarily complex models with lower computational requirements as the number of neurons are very low compared to the traditional algorithms [2]. This fact allows the algorithm to have both low latency and resource requirements without loss of accuracy. Also given the fact that the model parameters are lower in number, it creates an opportunity to completely map the algorithm to a pure hardware implementation [3]. In various devices that are used for time critical applications such as industrial robots and self-driving cars etc. processing speed demand has been increasing. An accelerator IP in an SOC can be extremely useful in real time applications where we need speed and guarantees on output latency. A software based implementation cannot be relied upon in such scenarios due to possibilities of OS induced delays. In this paper we present an optimized ASIC design for an ELM inference engine IP. In Section II we provide a brief overview of the extreme learning machine algorithm and also describe the various hardware implementations of ELM from academia. In section III we discuss in detail our proposed architecture along with an overview of the data flow. Section IV provides details of our synthesized design along with design constraints, key performance parameters of the design and proposal for NVM as an alternative for synapse. Section V presents a short description of possible enhancements that can be implemented to the current design.

II. Related Works

In this section we provide a brief overview of the ELM algorithm in order to have a better understanding of the data flow. We also provide a brief overview of some hardware implementations described in academic research.

A. Extreme learning Machine

Feed forward neural networks are widely used for modeling complex neural and artificial problems which are tedious to handle using conventional parametric equations solution. Generally, the learning speed is slower in case of feed forward neural networks when the training is based on gradient descent [5]. They take a significant time to converge as network parameters are required to be tweaked iteratively [6]. S. Tamura and M. Tateishi had demonstrated in their work that if single hidden layer neural networks had N hidden nodes and its hidden layer weights and biases are assigned randomly then this network can learn exactly N distinct input samples [7]. Based on this Yao [6] proved that hidden layer weights and biases can be randomly assigned if the activation function in hidden layer neurons is infinitely differentiable and weights of networks

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00094

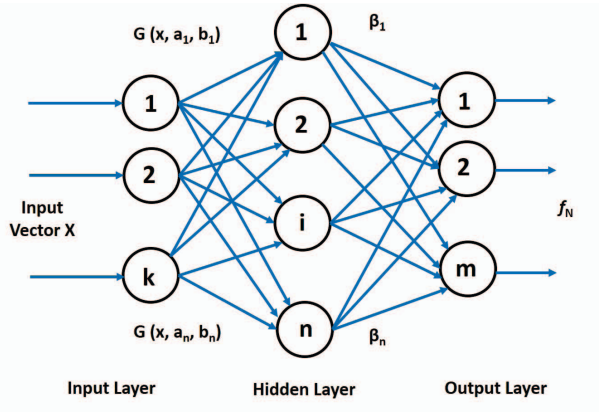

Fig. 1. Single Hidden Layer Neural Network used for ELM [4]

output layer can be analytically determined by finding inverse of hidden layer output matrix. This new training methodology led to the creation of ELM. The architecture for a basic ELM network is shown in Fig. 1. Suppose, there are N hidden nodes in the SLFN then it can be represented by (1). Here, G_i is defined using (2). The function g is defined as the activation function. In this work we have chosen sigmoid function for activation function (defined in (3)). For a given set of training data i.e. (x_i, t_i), where x_i and t_i represent training data vector and target output for i^{th} input respectively. The goal is to minimize the following training error also called cost function defined in (4). The training process is defined in following steps:

1) Assign hidden node weights and bias values randomly, for all i = 0,1,2.....N.
2) Find out hidden layer output matrix H.
3) Calculate the output weight vector as given in (5) , where H^{-1} represents the inverse of matrix H.

$$f_N = \sum_{i=1}^{N} (\beta_i G_i(x, a_i, b_i)) \tag{1}$$

$$G_i(x, a_i, b_i) = g(a_i.x + b_i) \tag{2}$$

$$g(x) = 1/(1 + e^{-x})) \tag{3}$$

$$e = \sum_{j=1}^{N} (\sum_{i=1}^{N'} \beta_i G_i(x_j, a_i, b_i) - t_j)^2 \tag{4}$$

$$\beta = H^{-1}.T \tag{5}$$

B. ELM implementations in literature

After the introduction of the ELM algorithm there have been several proposals in literature describing an accelerator design for the algorithm [8]–[14]. The most simple approach for hardware implementation aims at FPGA based deployment [8], [13]. Implementations have also been proposed that use synapses based on emerging NVM devices such as RRAM [12], nanowires [11], memristors [10] for the implementation. Basu et al. [9] described a spiking neural network based implementation. The currently proposed architecture is primarily inspired by the design presented in [8].

III. Proposed Architecture

Fig. 2 shows the top level architecture of design. The main blocks of the design are: (i) Input registers (ii) Hidden Layer PE (iii) Output layer PE (iv) Output Weight Memory (v) Output Registers. The functionality of each block is explained in detail in Section IV. Implementing training as part of the accelerator is huge in terms of area and power. Most of the neural networks implementations are trained only once till sufficient accuracy is achieved and used only for inference after that. Hence in this work, the classifier is trained off-line using MATLAB (algorithm discussed in [8]) to optimize area requirements and reduce complexity. The data precision for the accelerator was chosen to be 16 bit fixed-point (4 bits integer, 12 bits fractional). This helps in reducing complexity for all computation modules.

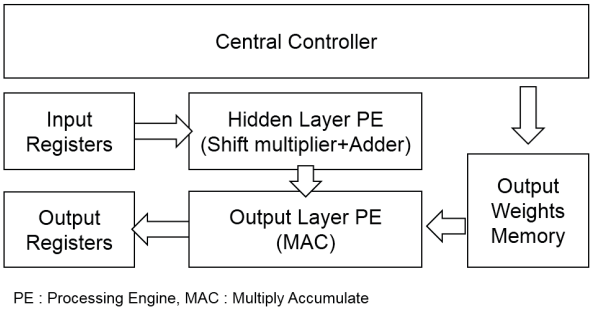

Fig. 2. Block diagram of proposed architecture

In order to generate weights for the network, we first randomly assign weights and biases for the hidden layer calculations. The random weights are generated in powers of 2 so that the multiplication of the hidden layer stage can be implemented using simple shift operations. This is possible since the data format used is fixed-point. Using the hidden layer weights and biases with all training dataset vectors, the complete hidden layer output matrix is generated. The output weight matrix can be obtained by optimizing the cost function or training error defined in (4) using the equation described in (5). These output layer weights are stored in output weight memory block to use in the Output Layer PE.

The data flow of the accelerator with all submodules is shown in Fig. 3. The operation of the accelerator can be divided into two stages: (i) the parallel stage which generates (M+1) dimensional vectors based on input vector X and weights and bias of input layer (M = number of hidden layer neurons) and (ii) the sequential stage which uses MAC based calculation for the output layer based on (M+1) dimensional vector generated from the previous layer.

To verify the performance of the proposed architecture, we simulated the design using the Pima Indians diabetes and image segmentation datasets available in the UCI repository

[15]. The datasets were divided in two sets of equal number of samples, first group was used to train the network and testing was performed on second set of dataset. Results of prediction accuracy are shown in Table I. Simulation results show that the predicted accuracy for Pima Indians diabetes and Image Segmentation dataset was 75% and 87.38% respectively. These results very closely match the prediction accuracy of the original ELM algorithm presented in [6].

TABLE I
ACCURACY RESULTS FOR THE SIMULATION OF THE PROPOSED ACCELERATOR

Datasets	Test Accuracy
Pima Indians Diabetes	75.00%
Image segmentation	87.38%

IV. IMPLEMENTATION

Fig. 3 shows the sub-modules of our current implementation. The input data vectors are loaded using the input registers. The hidden layer weights are pre-configured in the hidden layer PE based on training in software. The input data vector [X] is multiplied with the random weights for the hidden layer [W] by using shift operation. For each input vector x, this stage produces a vector H_j each with dimension [M+1] where M represents the number of hidden layer neurons. The output for each hidden layer neuron j is given by (6), where N is the dimension of the input vector x and G represents the activation function. These [M+1] dimensional vectors are passed through a multiplexer so that the parallel input can be sequentially given to later stages. Next stage comprises of neurons which adds up all the incoming vectors and applies activation function (sigmoid function is taken as activation function). The output of neuron stage is given to MAC unit which multiplies the input with the output weight matrix. MAC block implements a FSM which has exactly as many states as number of hidden neurons, so that it can process summation over the entire hidden layer output vector in a sequential way. One complete cycle of FSM transitions asserts *out_ready* flag to indicate completion of operation after which output is stored in the output register block.

$$H_j = \sum_{i=1}^{N} (w_{j,i} x_i) \tag{6}$$

A. Design Constraints

In order to benchmark our proposed design, we have synthesized the accelerator for a network of dimensions 81x18x1. The data precision is 16-bit fixed point as discussed earlier in section III. The RTL of design is simulated with Cadence Incisive tool and is functionally verified. The design is synthesized with 45 nm as well as 180 nm CMOS technologies. The accelerator is designed to operate at 10 MHz clock frequency. The design requires a voltage supply of 1.8V and 1V for 180 nm and 45 nm technology nodes respectively. We estimated

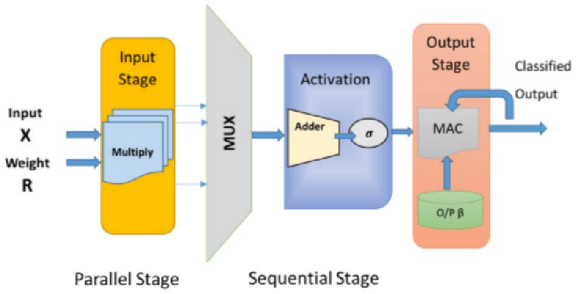

Fig. 3. Implementation of the sub modules

power at the best process condition i.e. FF, temperature = 125°C. Latency was estimated at worst process condition i.e SS, temperature = 125°C. These corners give us the worst case estimates of power and latency for the design. For the 45nm the delay numbers are also checked at slow process and low temperature i.e SS, temperature = -40°C. This is done to ensure that operation of the design is not affected by the temperature inversion effect [16].

B. Results

TABLE II
POWER, AREA AND PERFORMANCE RESULTS FOR THE IMPLEMENTED ELM ACCELERATOR

Technology	Area (μm^2)	Dynamic Power (μW)	Leakage Power (nW)	Response Time (ns)	Dynamic Energy / Op (pJ)
45nm	6033	13.269	396.93	504	6.69
180nm	84192	95.556	1089.267	672	64.21

Table II shows the static and dynamic power dissipation, area, achieved delay and dynamic energy per inference operation of our synthesized design for both technology nodes. It can be clearly seen that the design benefits from technology scaling. There is significant improvement observed in terms of power, area and latency. We can also observe from the area and power estimates. The floorplan and layout of the design are shown in Fig. 4 and Fig. 5 respectively. Table III presents a comparison of accuracy over diabetes dataset for previous implementations of ELM and other classification algorithms. These classifications algorithms were implemented on MATLAB with Intel Pentium 4 3.2 GHz CPU and 1 G RAM.

When the number of nodes in the neural network increase, the size of memory will also increase proportionally. However, using alternative memory architectures e.g. banking, butterfly etc. the overall area can be optimized.

V. PROPOSED ENHANCEMENT WITH EMERGING NVM TECHNOLOGY

From the floorplan of the design (Fig. 4) we can observe that ∼36% of the total area of the accelerator is occupied

TABLE III
COMPARISON OF CURRENT IMPLEMENTATION IN TERMS OF ACCURACY WITH OTHER ALGORITHMS

Classification Algorithm	Inference Accuracy	Hidden Nodes
ELM (This work)	75%	18
ELM (Double precision)	77.58% [6]	20
SVM	76.5% [17]	NA
RBF	75.9% [17]	NA
k-NN	73.44% [18]	NA
BP	76.34% [18]	20
EN-ELM	79.71% [18]	20

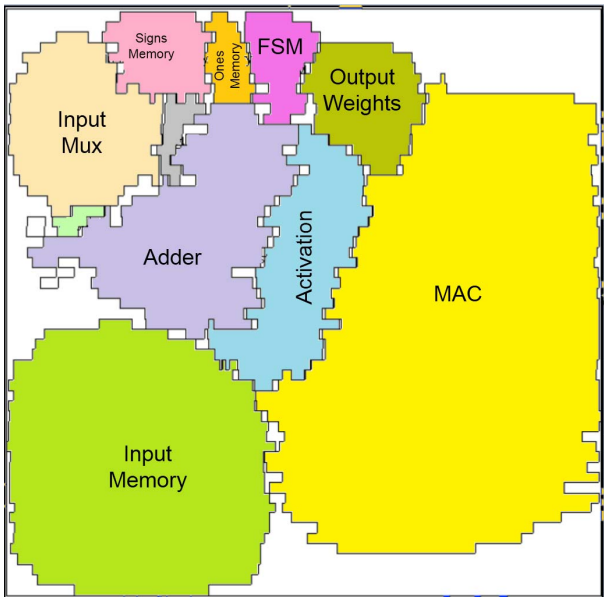

Fig. 4. Floorplan of ELM Accelerator IP

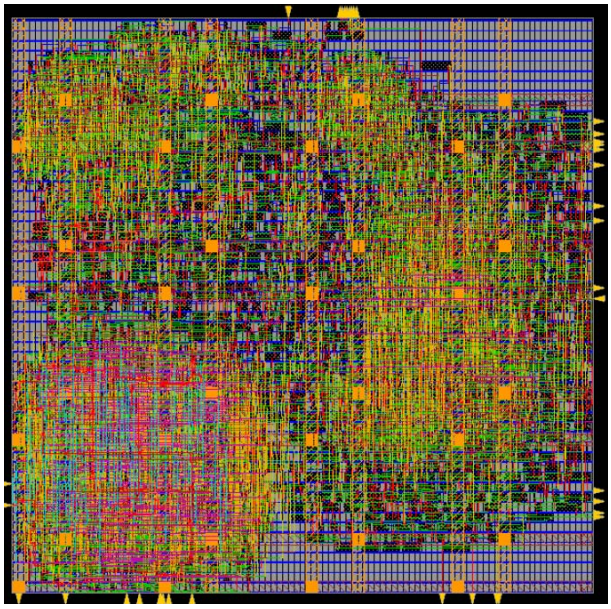

Fig. 5. Layout of ELM Accelerator IP

by memory cells. This indicates that further optimization in the circuit would require optimizing power and area of the memory. Another important observation that can be made with respect to functionality of the accelerator is that using CMOS based memory would lead to data loss during the power-down state and a high cost in terms of energy and delay is incurred for re-initialization of the accelerator with the weights. Conventional CMOS memory can be replaced with retention flops [19] or NVM based memory to make the accelerator resilient to this limitation [20]. The last decade has seen a rapid rise in innovation in the NVM technology as a result of emergence of a multitude of nano-scale that exhibit excellent NVM behaviour at very low power and feature size (OxRAM [21], CBRAM [22], PCM [23], STT-MRAM [24]). Many NVM-based designs have emerged [20] [25] which can potentially replace the CMOS-based retention flip-flops [26]. In our accelerator design, due to inference mode operations the memory would be primarily used in read mode. Hence for gaining the maximum area benefit, we propose use of the RRAM technology for replacing CMOS-based memory cells. In order to understand the impact of introduced RRAM based memory cells in the design we perform a comparative analysis for simulation of a large array using NVSIM [27]. NVSim, a circuit-level modeling tool for NVM performance, energy, and area estimation which supports various NVM technologies including STT-RAM, PCRAM, ReRAM, and conventional NAND Flash. For our current design, we used a RRAM bit-cell specification [28] as shown in Table IV. We derive estimates of leakage and area saving based on NVSIM simulations of SRAM and RRAM based memory macros of 2 MB at the 180 nm technology node. Based on this we scale the leakage and area of the CMOS-based cells used in the current design to estimate the improvement. Table V shows the comparison of area and leakage numbers of NVM memory (based on NVSIM) and shift register based memory. The shift register based memory of size 2 MB is created using the 180 nm library flip-flops while the NVM based memory is using NVSIM. We calculated the area and leakage power number based on the liberty files [29] available in 180 nm technology. In the shift register based memory excluded the decoder and MUX structures. From Table V we can observe that replacing the conventional CMOS-based memory with RRAM would give significant benefits in leakage (4.5x) and area (33x).

TABLE IV
RRAM DEVICE PARAMETERS USED IN NVSIM [28]

Parameters	Values
Resistance ON/OFF state (LRS/HRS)	100 kΩ / 10 MΩ
Resistance ON/OFF state (Read)	1MΩ / 10 MΩ
Read (Voltage/energy)	0.4V / 0.1 pJ
Reset (Voltage/pulse/energy)	2V / 10 ns / 0.6 pJ
Set (Voltage/pulse/energy)	2V / 10 ns / 0.6 pJ

VI. CONCLUSION

In this work, we proposed an ASIC implementation of neural network inference engine based on a modified version

TABLE V
LEAKAGE POWER AND AREA RESULTS COMPARISON NVM MEMORY VS SHIFT REGISTER BASED MEMORY

Module	Area (mm^2)	Leakage power (mW)
NVM	3.912	0.416
Shift registers	130	1.9

of the ELM algorithm. The proposed design can be used as an accelerator in a larger SOC for a defined application. We analyzed the impact of the modified algorithm on the learning performance for standard datasets as shown in Table I. We also demonstrated through synthesis on various technology nodes the benefits in terms of power, area and speed (Table II). Further we performed an analysis on the impact of replacing the current CMOS-based memory of the accelerator with RRAM based memory in terms of area and power (Table V). The inference time of the accelerator is 0.504μs at a power budget of 13.27μW with an area of $\sim$ 6000 μm^2 for the currently synthesized system at 45 nm. From this result, we can observe that such specialized accelerators can find wide usage in resource-constrained systems like IoTs where power and area are a critical criteria.

VII. ACKNOWLEDGMENT

The PI Dr. Manan Suri would like to acknowledge following research grants for partially supporting this research activity: DST-SERB-EMR, IIT-D FIRP, and MHRD IMPRINT.

REFERENCES

[1] C. D. Jiexiong Tang and G.-B. Huang, "Extreme learning machine for multilayer perceptron," in *IEEE Transactions on Neural Networks and Learning Systems, VOL. 27*. IEEE, 2016, pp. 809–820.

[2] V. Parmar and M.Suri, "Design exploration of iot centric neural inference accelerators," *ACM GLSVLSI*, 2018.

[3] S. Verma, N. Bhatia, S. Singh, , and M.Suri, "Low power neuromorphic hardware based multi-modal authentication system," *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 132–135, 2017.

[4] S. Sidler, B. Robert, M. Shelby, P. Narayanan, J. Jangy, lessandro Fumarolaz, K. Moony, Y. Leblebiciz, H. Hwangy, and G. W. Burr, "Large-scale neural networks implemented with on-volatile memory as the synaptic weight element: impact of conductance response," *Solid-State Device Research Conference*, vol. 46, pp. 440–443, 2016.

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 985–990.

[6] C.-K. S. Guang-Bin Huang, Qin-Yu Zhu, "Extreme learning machine: Theory and applications," in *Elsevier, Neurocomputing*. Elsevier, 2006, pp. 489–501.

[7] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three," in *IEEE Transactions on Neural Networks*. IEEE, 1997, pp. 251–255.

[8] A. L. Sergio Decherchi, Paolo Gastaldo and R. Zunino, "Efficient digital implementation of extreme learning machines for classification," in *IEEE Transactions on Circuits and SystemsII: Express Briefs, VOL. 59, NO. 8*. IEEE, 2012, pp. 496–500.

[9] A. Basu, S. Shuo, H. Zhou, M. H. Lim, and G.-B. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," *Neurocomputing*, vol. 102, pp. 125–134, 2013.

[10] C. Merkel and D. Kudithipudi, "A current-mode cmos/memristor hybrid implementation of an extreme learning machine," in *Proceedings of the 24th edition of the Great Lakes Symposium on VLSI*. ACM, 2014, pp. 241–242.

[11] Y. Wang, H. Yu, L. Ni, G.-B. Huang, M. Yan, C. Weng, W. Yang, and J. Zhao, "An energy-efficient nonvolatile in-memory computing architecture for extreme learning machine by domain-wall nanowire devices," *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 998–1012, 2015.

[12] M. Suri and V. Parmar, "Exploiting intrinsic variability of filamentary resistive memory for extreme learning machine architectures," *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 963–968, 2015.

[13] J. V. Frances-Villora, A. Rosado-Muñoz, J. M. Martínez-Villena, M. Bataller-Mompean, J. F. Guerrero, and M. Wegrzyn, "Hardware implementation of real-time extreme learning machine in fpga: analysis of precision, resource occupation and performance," *Computers & Electrical Engineering*, vol. 51, pp. 139–156, 2016.

[14] E. Yao and A. Basu, "Vlsi extreme learning machine: A design space exploration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 60–74, 2017.

[15] A. Frank and A. Asuncion, "Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california," *School of information and computer science*, vol. 213, 2010.

[16] W. Lee, Y. Wang, T. Cui, S. Nazarian, and M. Pedram, "Dynamic thermal management for finfet-based circuits exploiting the temperature effect inversion phenomenon," in *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM, 2014, pp. 105–110.

[17] G. Rätsch, T. Onoda, and K. R. Müller, "An improvement of adaboost to avoid overfitting," in *Proc. of the Int. Conf. on Neural Information Processing*. Citeseer, 1998.

[18] L. Nan and w. Han, "Ensemble based extreme learning machine," *IEEE Signal Processing Letters, VOL. 17, NO. 8, AUGUST 2010*, vol. 17, no. 8, pp. 754–757, 2010.

[19] H. Mahmoodi-Meimand and K. Roy, "Data-retention flip-flops for power-down applications," in *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, vol. 2. IEEE, 2004, pp. II–677.

[20] M.Suri, "Unconventional computing with emerging rram nanodevices," *IEEE International Conference on Nanotechnology (NANO)*, 2016.

[21] A. Chen, S. Haddad, Y.-C. Wu, T.-N. Fang, Z. Lan, S. Avanzino, S. Pangrle, M. Buynoski, M. Rathor, W. Cai *et al.*, "Non-volatile resistive switching for advanced memory applications," in *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*. IEEE, 2005, pp. 746–749.

[22] B. DeSalvo, E. Vianello, O. Thomas, F. Clermidy, O. Bichler, C. Gamrat, and L. Perniola, "Emerging resistive memories for low power embedded applications and neuromorphic systems," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium On*. IEEE, 2015, pp. 3088–3091.

[23] S. Lai and T. Lowrey, "Oum-a 180 nm nonvolatile memory cell element technology for stand alone and embedded applications," in *Electron Devices Meeting, 2001. IEDM'01. Technical Digest. International*. IEEE, 2001, pp. 36–5.

[24] W. Zhao, E. Belhaire, and C. Chappert, "Spin-mtj based non-volatile flip-flop," in *Nanotechnology, 2007. IEEE-NANO 2007. 7th IEEE Conference on*. IEEE, 2007, pp. 399–402.

[25] M.Suri, V. Parmar, G. Sassine, , and F. Alibart, "Oxram based elm architecture for multi-class classification applications," *IEEE International Joint Conference on Neural Networks, IJCNN*, 2015.

[26] J.-M. Portal, M. Bocquet, M. Moreau, H. Aziza, D. Deleruyelle, Y. Zhang, W. Kang, J.-O. Klein, Y.-G. Zhang, C. Chappert *et al.*, "An overview of non-volatile flip-flops based on emerging memory technologies," *Journal of Electronic Science and Technology*, vol. 12, no. 2, pp. 173–181, 2014.

[27] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[28] S.-S. Sheu, M.-F. Chang, K.-F. Lin, C.-W. Wu, Y.-S. Chen, P.-F. Chiu, C.-C. Kuo, Y.-S. Yang, P.-C. Chiang, W.-P. Lin *et al.*, "A 4mb embedded slc resistive-ram macro with 7.2 ns read-write random-access time and 160ns mlc-access capability," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*. IEEE, 2011, pp. 200–202.

[29] Synopsys, "Library compiler reference manual volumes 1-3," 2007.