Design and Testing Considerations of an In-Memory AI Chip

Marcia Golmohamadi, Ryan Jurasek, Wolfgang Hokenmaier, Don Labrecque, Ruoyo Zhi, Bret Dale, Nibir Islam, Dave Kinney, Angela Johnson

***Abstract*—In-memory computing is a propitious solution for overcoming the memory bottleneck for future computer systems. In this work, we present the testing and validation considerations for a programmable artificial neural network (ANN) integrated within a phase change memory (PCM) chip, featuring a Nor-Flash compatible serial peripheral interface (SPI). In this paper, we introduce our method for validating the circuit components specific to the ANN application. In addition, high-density in-memory multi-layer ANNs cannot be manufactured without testing and repair of the memory array itself. Therefore, design for testability (DFT) features commonly used in commodity or embedded memory products must be maintained as well. The combination of these two test/characterization steps alleviates the need to test the actual inference functionality in hardware.**

***Index Terms*—In-memory Computing, DFT, Scan Chain, PCM**

I. Introduction

In the von Neumann architecture data must be sent back and forth between the memory and the processor. Data movement is very expensive in terms of bandwidth, energy and latency and speed of data transfer can not catch up with the increased rate of processor's speeds. This problem has become particularly critical in large deep learning neural networks, where the amount of data movement is huge. To overcome this increasing processor-memory performance gap, memory system architectures should be organized in different ways to make them more intelligent. One solution is referred to as in-memory computing, which exploits the physical properties of memory devices for both storing and processing [1]. The research toward such a departure from classic processor-centric von Neumann architecture has for some time focused on the analog properties of the memory cells. Especially, memristor arrays have received particular interest in the literature [2]. However, high power DACs/ADCs circuits along with non-linear and variable conductance response make this approach less than ideal [3]. On the contrary, digital in-memory computing paradigms only use ON-state and OFF-state which are are much more reliable than the intermediate values used in analog frameworks.

With the anticipated high-volume production of hardware neural networks in the near future, testing strategies specific to hardware neural networks is a new topics that is largely uninvestigated. The aim of this paper is twofold. Firstly, we want to give an overview of the test and reliability considerations of hardware implemented neural networks. Secondly, we present the DFT approach based on partitioning a module-based design for a fully functional and programmable AI chip that is consistent with a wide variety of memory technologies. To the best of our knowledge, this is the first work that studies modular test consideration in a metamorphic chip. The memory array used in this chip is a non-volatile phase change memory (PCM) [4].

The paper is organized as follows. In Section II, we first provide a brief introduction to neural networks and neuron structure. Then, we go through our proposed architecture for implementation of a neural network inside the memory chip. Section III provides a review of current approaches toward testing and verification of neuromorphic chips. In section IV, we explain the design for test features of both the PCM array and the neuron circuitry of the chip. In section V, we describe how to verify the inference chip and how to program the chip as an inference machine and finally in section VII, the summary and future steps will be discussed.

II. System Overview

A. Background

An ANN is made up of interconnected neurons and contains an input layer, an output layer, and one or more hidden layers. Fig. 1 shows a fully connected Multi-Layer Perceptron (MLP) ANN and Fig. 2 represents the functionality of each node. The node circuit is comprised of a multiply-accumulate (MAC) unit and an activation function. Each input is multiplied by a unique weight (w_1 through w_N) and these products are accumulated. Finally, the bias is added to the accumulated result. Subsequently, the 32-bits accumulated values are passed to an activation function (AF) in order to add non-linearity to the network and also to re-scale the MAC output to 8-bit integer representation. As it is presented in Fig. 1, a MLP neural network can be realized as the connectivity between neurons. In these types of neural networks, the basic operation of a neuron is multiplication and accumulation, so the MAC units can be reused to mimic a range of neural network sizes and types by programming the connectivity between the units. Furthermore, the activation function is executed as a re-configurable look-up table, which adds extra flexibility to the design.

B. System architecture

Fig. 3 illustrates a high level overview of the proposed AI inference architecture. It consists of four interconnected blocks: a non-volatile memory array, an inference control circuit (ICC), node circuits, and a Layer Buffer (LB). All the information of a neural network are written in the memory

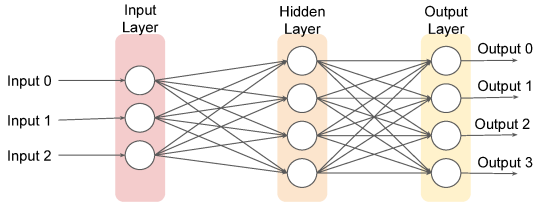

Fig. 1: A fully connected neural network

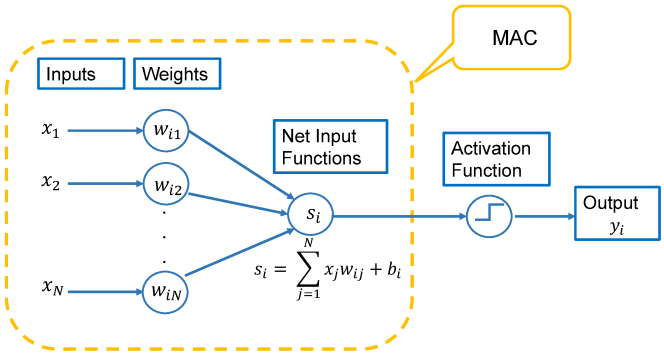

Fig. 2: A neuron of a NN

array. The ICC block decodes the ANN definition information written in the array and accordingly, controls execution commands to carry out the inference. The node circuit contains MAC circuits and activation functions. The summary of implementation of the inference phase of a neural network can be described as follows:

1) At chip power-up, the ANN definition, which includes weights, biases and connections, is written into the array, so the ICC knows the commands and order of their execution in addition to the address of all needed weights, biases and AF data in the array.
2) When needed, the processor issues an inference command to the memory that contains the input data and the identifier for selecting one ANN, in case several neural networks are stored in the memory.
3) When the ICC has all the required information, the ICC will start calculating the nodes' outputs. The weights, biases and the activation function definition are read from the memory array to the node circuit via a read command issued from the ICC. The intermediate data at each layer is stored in the LB for use in the next layers.

III. Testing Strategies in Neuromorphic Computing

A major challenge with future technology scaling is the growth of fault rates, including both permanent (hard errors) and temporary faults (soft errors). Neural networks have some degree of robustness against failure in the activation or weight data. However, there are few works that have studied the extent of fault tolerance in neural networks and its dependency on the training algorithms or the structure of the network. It has been

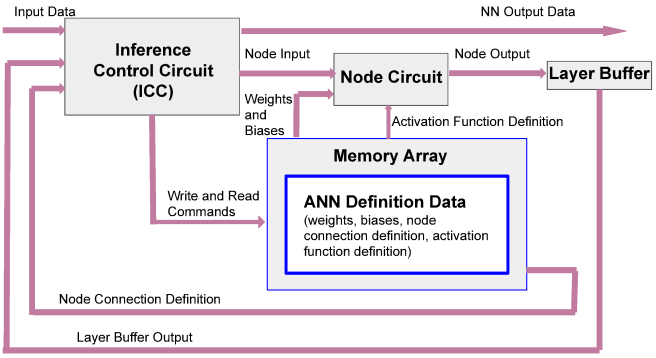

Fig. 3: Flow of implementation of a neural network inside memory chip

shown that in a feed-forward neural network, as the single-bit-failure rate increases to 20%, the average accuracy drops from 92.64% to 39.4% [5]. To solve hard failures in neuromorphic computing, it has been suggested to detect the failed locations in the memory array [5] or in the processing elements [6] and to generate a failure map. Then, based on the failure map, they set all the weights or activation in faulty places to zero. But, in order to keep the accuracy, the network should be re-trained based on the pruned connections. The conventional method to deal with soft errors is ECC blocks, which has been said adds about 10% to 20% power and space overhead to the chip. However, in neuromorphic computing, ECCs might have to be minimized or removed depending on the design of the computing system. In addition, fault tolerance inherent in neural networks can help to overcome the temporary errors by designing very simple error correction circuits in the chip, like implementation of parity bit error detection and setting the erroneous data to zero. In case of a high error rate, if this technique is not accompanied with compensating training algorithms, it can cause a significant accuracy drop of the network [7]. Here we provide a summary of a paper that has introduced a technique to deal with soft errors in neuromorphic computing. A low overhead error detection and correction approach for multilayer artificial neural networks is proposed in [8] by adding checker neurons to the network during training. This work adds checker neurons to the hardware and compares the checker output with sum of all neurons in a hidden layer, if there is a mismatch, it issues an error flag. In this case, the corresponding hidden layer is bypassed and the network will be retrained without the faulty hidden layer.

IV. DFT Methodology in the Proposed Neuromrphic Chip

In this section, first the testing and repair capability of the PCM array as a stand-alone commodity memory chip, will be discussed and subsequently, the developed methods for post-silicon validation of ANN circuit components will be presented.

A. Characterization and validation of memory array

PCM is a resistive memory technology that is based on the transition between the crystalline and amorphous phases of specific materials by the application of suitable electrical pulses. Furthermore, the use of DFT methodology in the chip, which allows flexibility in the control of many of the critical timings and voltages, is discussed. PCM is a type of non-volatile memory with some properties comparing favorably to Flash Memory. Essentially, the PCM cell consists of a chalcogenide layer (i.e. GST), a heater and a select transistor. The structural state of the cell can change from a crystalline phase with low resistance to an amorphous phase by applying heat in the form of an electric pulse. The low resistance crystalline state, also known as SET state, is used to identify a cell with a logic value '1', whereas the high resistance amorphous state is known as RESET state where the logic value it represents is a logic '0'. During the RESET operation, which is the operation that logic '0' is written to the cell, the temperature of the device is increased a little above the melting point and then suddenly quenched to cool it. The rate of cooling and the RESET pulse width are critical parameters for the development of the amorphous layer. Whereas during the SET operation, which is the operation that logic '1' is written to the cell, the temperature of the PCM cell is raised above the crystallization temperature but lower than the melting point, then cooled slightly slower to allow the formation of a crystalline layer. The SET pulse width along with its rise time and fall time are additional parameters that need to be characterized [9]. In what follows, we discuss characterization of these parameters and replacing defective cell arrays with the redundant elements in our PCM design.

The proposed PCM array design is the first design to combine 40nM front-end processing technology with the PCM processing to create a non-volatile Serial Port CMOS memory. The design is done with extensive use of DFT methodology, which allows flexibility in the control of many of the critical timings and voltages used internally. The primary DFT features are accessed through test modes which are loaded into the serial port of the scan chain using a special command sequence. These Test Modes are targeted for several primary areas such as:

1) Manufacturing Test
   - Redundancy testing and implementation
   - Optimization of the SET and RESET pulse shapes
   - Optimization of the Sense Amplifier
2) Diagnostic Tests
   - Test modes to view internal chip voltages and waveforms through an external pad. These tests modes can be performed by issuing SPI commands utilizing the external PADs and by monitoring the "SO" PAD.

Our programmable non-volatile PCM device includes a memory array of addressable memory cells and multiple redundant memory cells for replacing defective memory cells. The core array is a 2-Mb tile with 1k wordlines and 2k bitlines. In addition, six wordlines and 64 bitlines are used for redundancy [4]. Before reading/writing data from/into the array, the data passes through the redundancy substitution block. The redundant memory circuit matches the addresses of the defective rows or columns with redundant rows and columns and subsequently routes the data to the redundant memory cells instead of the defective memory cells. In addition, the input data can be scanned into the redundancy block to verify read commands and redundancy modules. Developing new PCM materials and devices requires characterizing a variety of parameters including characteristics of SET/RESET pulses. The design also provides test modes to optimize the amplitude and the shape of SET/RESET pulses that forced through the cell to heat the material (Fig. 4).

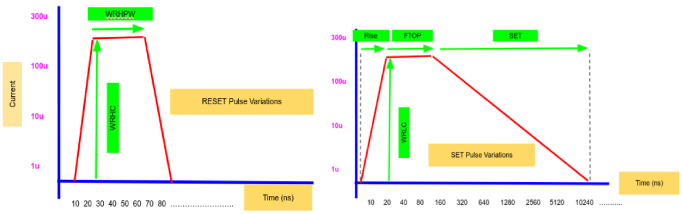

Fig. 4: SET/RESET optimization parameters during the manufacturing test

Furthermore, sense amplifier design is equipped with test modes to control the reference resistor to cope with the variations of the PCM device SET/RESET resistance values.

B. AI computation circuit test

To provide the design with flexibility and observability characteristics, a scan chain is developed to monitor the signals inside a node circuit (Fig. 5). In scan chain testing methodology, the sequential logic circuits are divided into a series of combinational logic blocks. This approach provides controllability of inputs and observability of outputs of the combinational logic blocks [10]. The test vectors are fed into the chip through the "SI" input pin. In what follows, the test sequence of the LB and all the computation units of a neuron (Fig. 5) will be discussed and then the whole online scan test is summarized.

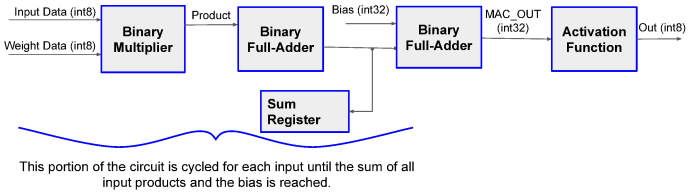

Fig. 5: Scan chain of a neuron (node circuit)

LB test: The LB is an array of latches that holds the intermediate neuron activation values during each layer of inference. It supports 8-bit parallel write and serial read. During loading test vectors, the test modes for the scan in

data should be chosen appropriately such that the data from scan in will be written into the LB. Then, the data from the scan chain is written into the LB. By issuing a read command the LB contents can be scanned out bit by bit. A block diagram of the online scan testing of the LB is presented in Fig. 6.

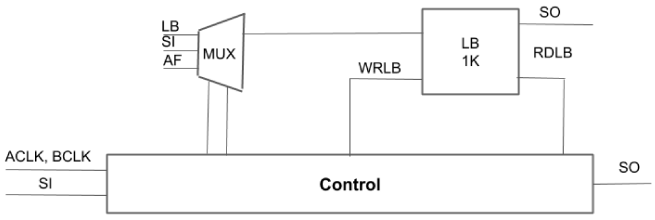

Fig. 6: Block diagram of on-line scan chain testing of the LB

The timing diagram of the required sequence of scan chain commands for the LB verification is shown in Fig. 7.

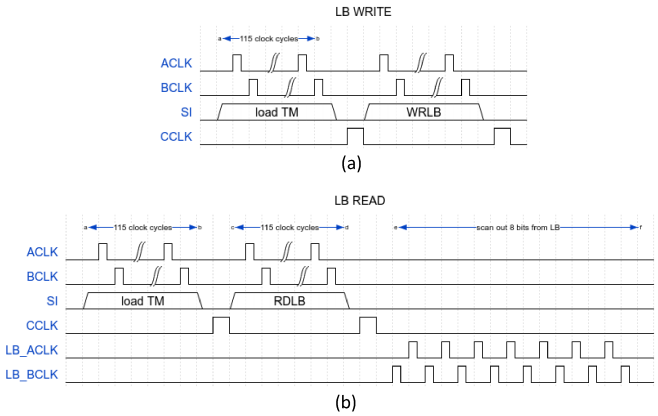

Fig. 7: Timing diagram of sequence of the on-line scan chain testing of the LB (a) Timing diagram of writing into LB from SI (b) Timing diagram of reading contents of layer buffer

The MAC blocks are the core computation units of our proposed neuromorphic chip design. There are 4 MAC blocks on the chip that allow executions of four neurons simultaneously. The multiplication and accumulation procedure begins with reading 32-bit array data. This data gets divided into 4 MAC. To perform the multiplication of 8-bit array data and 8-bit LB data, LATCH-MULT signal is generated 8 times during reading 8-bits from the LB serially. Then, this product gets accumulated with the previous product results. To test MAC modules, values of the LB registers should be scanned in as described above. Then, test modes should be set such that one of the MAC inputs is fed from the scan chain data instead of the array and the other input flows from the LB to MAC blocks. This mode allows the MAC block to be tested individually, independent of the memory array data. In addition, there are a couple of test modes that represent different combinations of input and weight data (i.e. (a) signed input data multiplied with signed weight data, (b) signed input data multiplied with unsigned weight data, (c) unsigned input data multiplied with unsigned weight data and (d) unsigned input data multiplied with signed weight data). It should be noted that for testing each MAC block, there is a unique test mode. Subsequently, multiply and accumulate commands will be sent to MAC. Finally, by reading the result off-chip, the MAC output can be monitored. The control lines of MAC scan chain are presented in Fig. 8. The sequence for testing MAC

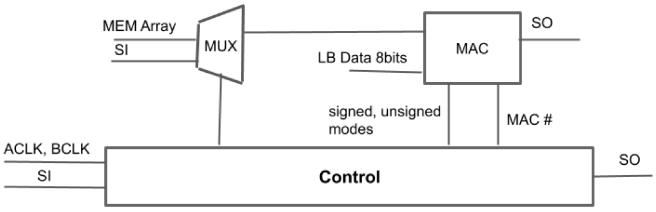

Fig. 8: Block diagram of on-line scan chain testing of MAC.

functionality can be summarized as follows:

1) Write data into the LB
2) Set Test modes such that scan chain data is chosen over memory array data for feeding into MAC
3) Reset Multipliers and Accumulators
4) Scan in LATCH-Mult command
5) Scan in Latch-ACC command
6) Read 32-bits out of Accumulator
7) Repeat this sequence for all MACs and also for all different modes of multiplication

The next block in executing a neuron in a neural network is the activation function (AF). In modern deep neural networks, there are many AFs that have been developed over the years. The proper choice for AFs can improve the learning of the patterns in data. However, there is no rule of thumb for choosing the best activation function The research toward finding more efficient AFs is still in progress. For example, for a while “sigmoid” and ”tanh” were the only activation functions used in neural networks. But due to several problems like vanishing gradient and slow convergence, other variations of these two AFs were introduced. Rectified linear units (ReLU) gained popularity due to its simplicity, providing better performance and higher convergence rate. ReLU is the most widely used AF in current architectures, especially CNN models. But the ReLU sometimes causes some neurons to be dead, thereby slowing down learning. To resolve the dead neuron issues, the leaky ReLU was proposed, which introduces some small negative slope in the negative side of input, to keep the weight updates active during the entire propagation process. Except for the leaky ReLU, there are three more variants of the ReLU. These variants have added a little more complexity to regular ReLU function to improve its performance. Exponential linear units (ELU) and ”SWISH” are two other types of AF that have been introduced after ReLU. In short, there are so many types of AFs in current neural network architectures, and it is very likely that new functions will be introduced in the future. So, designing the AF block as a programmable look-

up table (LUT) that is able to keep up with all these changes, makes much more sense [11]. The proposed AF design is capable of achieving any monotonic activation function by programming different register values into it. To program the activation function, a specific test mode has to be first loaded and then all the LUT register values can be scanned in through the scan chain input to the AF. After all the AF register values are loaded, the test mode has to be switched back to the normal operation so that the values are preserved in the registers in the AF. During the normal operations, the values can be scanned in into the AF block and be scanned out to test the its functionality. The AF is a flow through device, where the output instantly reflects the changes of the input without other control signals. To test the AF, we built a testbench that traverses from the smallest AF register value to the largest register value as input to AF, and then compares the output with the simulation result.

V. AI Chip Verification

During execution of ANNs in the hardware, a suitable number format should be chosen so that both accuracy and cost conditions are met. For successful implementation of deep neural networks on resource constrained devices, we have moved to a network with 8-bit integer-only arithmetic [12], [13]. To program the chip, the NN model should first be trained. We used TensorFlow, which is a Google machine learning framework, to train our model. Since training with low precision numbers is challenging due to the gradient computations during the back-propagation phase, the training is done using a 32-bit floating point. To achieve more accuracy for the quantized inference model, quantization-aware training can be performed using TensorFlow-lite [12]. To implement the inference framework in our hardware, the network should be quantized into 8-bit integers. The design has the flexibility to accommodate either symmetric or asymmetric quantization methods, as none of them have proved to be better than the other [14]. It has been verified that reducing the inference model sizes from 32-bit to 8-bit will result in minimal accuracy losses [12]. Our simulated quantized inference scheme, which is consistent with the computation flow in the proposed hardware model showed a drop of less than 2% accuracy for an image recognition task using MNIST dataset in comparison with floating point inference model results.

To program the chip, the complete neural network definition is transferred to the memory chip and stored in the memory array. A protocol is defined to store the network information in a consecutive manner. This protocol only needs the start address to fully execute a deep neural network. It currently allows for up to 1024 neurons per layer and up to 1024 inputs per neuron, with no limit on the amount of layers. For a non-volatile memory such as PCM this needs to be performed only once for a given solution, whereas a volatile memory will need to be initialized after each complete power-down. This design is also capable of providing sparsity in deep neural networks, which is a compression technique for the reduction of the model size [14].

TABLE I: Impact of drpout and a simple ECC on robustness of a feedforward NN

NN architecture	FF-2 w/o ECC	FF-2 w ECC
BER = 0, w/o dropout	88%	88%
BER = 0.01, w/o dropout	71%	85%
BER = 0.001, w/o dropout	86%	88%
BER = 0, w 0.4 dropout	87%	87%
BER = 0.01, w 0.4 dropout	79%	85%
BER = 0.001, w 0.4 dropout	86%	87%

VI. Future work

Still there are a number of different ways that the work in this paper could be extended. These proposed paths for future research can be classified as follows. To explore training algorithms that are more robust to faults in the hardware. These fault can be memory faults or processing units faults. By estimating the error rate in both memory array and in the processing units, some appropriate training algorithms can be used to moderate the fault effects. Specifically, Droput, which is a regularization technique that has developed to solve over-fitting problems in the training process and randomly removes some connections or weights, can be applied during the training. We are going to set the dropout rate based on the knowledge about the possible error rate in our hardware and study its impact on the fault tolerance of different network structures. Another viable path for extending this work is to incorporate a simple parity based ECC technique in our design to concurrently detect the soft errors and minimize their impact on the neural network performance. The soft errors are caused by transients like power-supply voltage spikes, thermal effects, and man made static. These errors are called soft because they do not damage the physical function of a cell for ever. In this simple on-chip ECC method single errors can be detected and instead of correcting the faulty bits, the neural network parameters (weights and biases) with faulty bit can be set to zero. In the following several neural network models have been examined to explore the effects of dropout and parameter nulling on the accuracy of the neural networks with faulty parameters. Table I summarizes the comparison of robustness a FF-NN with and without ECC against faulty bits. To evaluate the effect of faulty bits on the NN accuracy, the trained parameters have been contaminated with errors having the rate of 0.01 and 0.001. The Fashion-MNIST database is used for this paper. It includes 60000 training images and 10000 test images of size 28×28. The 3-layer FF-NN is trained and quantized using a TensorFlow Lite framework. In addition, the effects of dropout regularizer technique on the robustness of NN has been studied. Based on the results presented in Table I, the improvement brought by the error detection and mitigation method during inference is more significant than dropout technique. However, impacts of the dropout on the performance of distorted NN is noticeable. For instance, adding BER of 0.01 to the parameters reduces the accuracy from 88% to 71%. This issue can be mitigated by implementing of the dropout with during the training phase, as the accuracy of 79% has been achieved in this case. It should

be notified that this improvement has been achieved only by a small change in the training method without adding any time and space overhead to the hardware.

VII. Conclusion

In this work, we have introduced hardware development of a fully digital in-memory inference AI design inside a PCM chip and the DFT methodology to characterize, program and verify the chip. We also have provided an overview of the challenges faced by hardware implemented neural networks in terms of reliability and test. The provided DFT features in chip allow traditional testing and repair of the memory array, while also giving access to the added functional blocks of the ANN for validation and performance evaluation. Furthermore, The components for the system level integration including a protocol to store and execute the network topology in memory and programming of the artificial neural network have been presented.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Phase II project, Grant No. 1831151.

References

[1] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, pp. 333–343, 2018.

[2] I. Giannopoulos, A. Sebastian, M. Le Gallo, V. Jonnalagadda, M. Sousa, M. Boon, and E. Eleftheriou, "8-bit precision in-memory multiplication with projected phase-change memory," in *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2018, pp. 27–7.

[3] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[4] W. Hokenmaier, D. Labrecque, R. Jurasek, V. Butler, C. Scoville, A. Willey, S. Loeffler, Y. Li, and S. Sharma, "A 90nm 32-mb phase change memory with flash spi compatibility," in *2014 IEEE 6th International Memory Workshop (IMW)*. IEEE, 2014, pp. 1–4.

[5] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[6] J. J. Zhang, T. Gu, K. Basu, and S. Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *2018 IEEE 36th VLSI Test Symposium (VTS)*. IEEE, 2018, pp. 1–6.

[7] M. Qin, C. Sun, and D. Vucinic, "Improving robustness of neural networks against bit flipping errors during inference," *Journal of Image and Graphics*, vol. 6, no. 2, 2018.

[8] S. Pandey, S. Banerjee, and A. Chatterjee, "Error resilient neuromorphic networks using checker neurons," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 135–138.

[9] M. G. Mohammad, "Fault model and test procedure for phase change memory," *IET computers & digital techniques*, vol. 5, no. 4, pp. 263–270, 2011.

[10] H. Al-Asaad and P. Moore, "Non-concurrent on-line testing via scan chains," in *2006 IEEE Autotestcon*. IEEE, 2006, pp. 683–689.

[11] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[12] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[13] M. Imani, M. Samragh, Y. Kim, S. Gupta, F. Koushanfar, and T. Rosing, "Rapidnn: In-memory deep neural network acceleration framework," *arXiv preprint arXiv:1806.05794*, 2018.

[14] N. S. Sohoni, C. R. Aberger, M. Leszczynski, J. Zhang, and C. Ré, "Low-memory neural network training: A technical report," *arXiv preprint arXiv:1904.10631*, 2019.