Special Session: Reliab
Harc

Shamik Kundu[1], Kanad Basu[1], Mehdi Sadi[2], Twisl
[1]Department of Electrical & Computer Engineering,
[2]Department of Electrical & Computer Engineering, A
[3]Department of Electrical & Computer Engine

***Abstract*—Artificial intelligence (AI) and Machine Learning (ML) are becoming pervasive in today's applications, such as autonomous vehicles, healthcare, aerospace, cybersecurity, and many critical applications. Ensuring the reliability and robustness of the underlying AI/ML hardware becomes our paramount importance. In this paper, we explore and evaluate the reliability of different AI/ML hardware. The first section outlines the reliability issues in a commercial systolic array-based ML accelerator in the presence of faults engendering from device-level non-idealities in the DRAM. Next, we quantified the impact of circuit-level faults in the MSB and LSB logic cones of the Multiply and Accumulate (MAC) block of the AI accelerator on the AI/ML accuracy. Finally, we present two key reliability issues – circuit aging and endurance in emerging neuromorphic hardware platforms and present our system-level approach to mitigate them.**

***Index Terms*—Machine learning, deep learning accelerator, neuromorphic computing, reliability**

I. Impact on DRAM Faults on DNN Accelerators

Deep Neural Networks (DNNs), with their ever increasing computing power are gaining momentum over classical machine learning and computer vision algorithms. As a result, DNNs are being extensively deployed across real-time data driven applications on resource constrained Internet-of-Things (IoT) edge devices. Conventional CPU architectures tend to lose out on implementing the computational complexity of state-of-the-art DNNs, which propelled the development of cost and energy efficient application-specific neural network accelerators. Both industry and academia responded to this emerging community of DNN accelerators by developing several purpose-built inference hardware [1]–[3]. Google's Tensor Processing Unit (TPU) is one such accelerator that achieves 15–30× higher performance and 30–80× higher performance-per-watt over traditional CPUs and GPUs [1]. These custom built DNN accelerators find their application in the domains of computer vision, multimedia processing, graph analytics, and search. With this widespread proliferation of DNNs, researchers have focused on performance optimization and energy-efficiency of such hardware architectures. Even though DNNs are presumed to be resilient against errors by virtue of their inherent fault tolerant capabilities, the threshold of such resiliency can be easily inflicted with bit-level hardware faults, leading to graceless degradation in classification accuracy of the DNN [4].

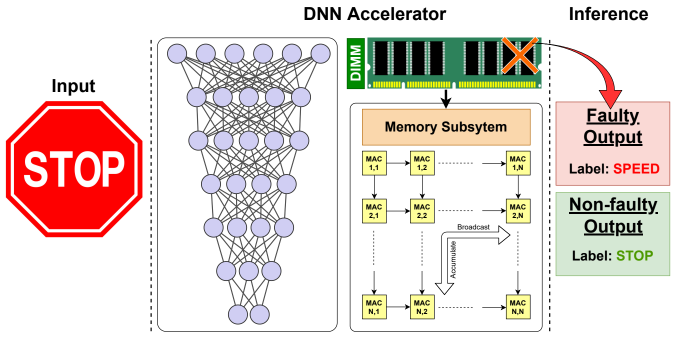


Figure 1: Fault manifestation in DRAM leading to misclassification in a mission-critical DNN accelerator.

To execute an inference network at the edge, considerable amount of on-chip memory is required to store millions of trained network parameters, input activations and intermediate feature maps. For this purpose, energy restricted DNN accelerators utilize Dynamic Random Access Memory (DRAM) as their primary memory subsystem, owing to its low access latency and high density architecture. However, due to various non-idealities associated with the access transistor, the charge stored in the DRAM cells leak away over time, engendering bit-level faults throughout the memory. In order to alleviate this, a DRAM is periodically replenished using an implicit background refresh operation, which contributes to a significant amount of DRAM energy overhead as well as performance bottleneck in the DNN accelerator. Existing research to alleviate this high refresh rate of DRAMs have focused on utilizing sub-optimal refresh rates; at the expense of bit-flip faults that are highly dependent on temperature and variable retention times of each cell in the structure. Such faults manifested in the structure adversely impact the classification of the network, as demonstrated in Figure 1. Since DNN accelerators are often deployed in high-assurance environments, *e.g.*, self driving vehicles for enhancing the autonomous driving dynamics, smart sensors in aerial surveillance vehicles like drones, analyzing the impact of these DRAM faults is highly imperative to avoid catastrophic circumstances.

A. DRAM Basics

DRAM is the primary choice for main memory in most computing platforms ranging from large scale data centers to miniature edge devices, due to its high density, longevity, and low cost. The basic element of the 2D-DRAM array is a DRAM bitcell, which consists of a single capacitor and an access transistor. Depending on the charge in the DRAM cell capacitor, a binary data value is identified as 1 when the capacitor is fully charged, or 0 when it is in a discharged state. However, DRAM capacitor loses charge over time due to various factors predominantly related to the non-ideality of the access transistor like sub-threshold leakage and gate-induced drain leakage. As a result of this, faults in the form of bit flips start to manifest in the data bits after a certain time interval (known as retention time). Therefore, the charge needs to be refreshed periodically to preserve the integrity of the data. The leakiest cell in the entire DRAM array determines the worst-case retention time, which is usually 64ms in case of commercial DRAM modules. Even though high refresh rates are imperative to maintain correctness, they also adversely impact on the overall energy efficiency and performance. Energy efficiency declines due to the periodic activation of individual rows as well as the increase in energy consumption due to a longer execution time.

Existing research to alleviate the high refresh rate of DRAMs have focused on utilizing sub-optimal refresh rates, approximating the device performance. One of the earliest work in this domain, Flikker, partitions the application data into critical and non-critical parts, injecting errors in the non-critical portion at higher refresh intervals [5]. Sparkk, a DRAM approximation framework refreshes the most significant bits at a higher refresh rate than the least significant ones [6]. Other DRAM approximation schemes have also been proposed, that reduces DRAM energy consumption at lower refresh rates [7]–[9]. A quality configurable approximate DRAM has been proposed, that utilizes the concept of critical and non-critical data partitioning to allocate data in multiple quality bins, considering the property of variable retention times exhibited by DRAM cells [10]. However, the impact of the errors on high assurance DNN accelerators has not been well explored at high DRAM refresh intervals, which drives us to analyze the reliability of such high assurance architectures.

B. *DNN Accelerator and Its Reliability*

In recent years, DNNs have acquired a meteoric rise in various spheres of life due to their use of sophisticated mathematical modeling to process data with high complexity parameters. To meet the extreme compute requirements of these compute-heavy DNN algorithms, a number of DNN accelerators have emerged over the past decade. Most DNN accelerators in practice utilizes DRAM as the main memory subsystem. For example, well known DNN accelerators such as Google TPU [1], Eyeriss [2], NVIDIA Jetson Dev Board [11], Google Coral Edge TPU [12], and Intel MyriadX VPU [13] consists of 8 GB, 1 GB, 4 GB, 1 GB, and 2 GB of DRAM, respectively. Recently, a new class of sparse DNN accelerators has emerged, such as Eyeriss v2 [14], NVIDIA Ampere [15], Intel Keem Bay VPU [16], *etc.*, that accelerates the performance of sparse matrix convolution in the inference. These accelerators leverage the sparsity in a tensor graph and therefore, skip certain computations during the inference, based on the bitmap encoding of the tensors. In these sparse accelerators, DRAM reliability is extremely critical, which can lead to subverting the accuracy of the DNN accelerator.

With the extensive deployment of DNN accelerators in a wide gamut of applications, researchers have analyzed the impact of faults in various inference hardware. To explore the correlation between bit error rate and model accuracy, permanent faults are injected in the memory elements of a customized DNN accelerator [17]. The impact of single bit soft errors in the memory on the network performance is explored in [18], which further proposed a bit-flip resilience optimization method for DNN accelerators. Memory faults are induced in the Autonomous Driving System (ADS) of a vehicle and the corresponding resilience of different modules are examined [19]. The susceptibility of the architecture under single event upsets on the datapath of an accelerator is analyzed on multiple Convolutional Neural Network (CNN) models [20]. The safety critical bits in a machine learning system were identified by inducing soft errors in the network datapath and evaluating them on eight DNN models across six datasets [21]. A formal analysis on the impact of faults in the datapath of an accelerator has been illustrated using Discrete-Time Markov Chain (DTMC) formalism [22]. The intense performance penalty in a systolic array-based DNN accelerator has been demonstrated by inducing manufacturing defects in the datapath of the accelerator [4], [23], [24]. However, fault characterization of DNN accelerators due to device-level non-idealities in the DRAM cells has not been well explored.

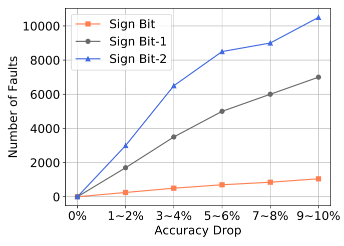
C. *Fault Characterization to Estimate Network Performance*

In order to analyze the impact of DRAM faults on the performance of the DNN accelerator, we implement Multilayer Perceptron (MLP) on two different datasets —MNIST and Fashion-MNIST. The detailed network configuration of the MLP is provided in Table I. We consider Google's Tensor Processing Unit as the baseline DNN accelerator, having 8GB of dual-channel DDR3 DRAM as the main memory subsystem. The trained weights from each network are extracted and quantized to 8 bits to be stored in the DRAM, similar to [4]. Subsequently, the trained weights from the DRAM are mapped on to the inference hardware. MNIST furnishes a baseline classification accuracy of 97.28%, whereas Fashion-MNIST manifest an accuracy of 88.17% on MLP. Errors in the form of bit-flips are induced throughout the DRAM structure, following which the application-level fault characterization of the accelerator is analyzed in this section.

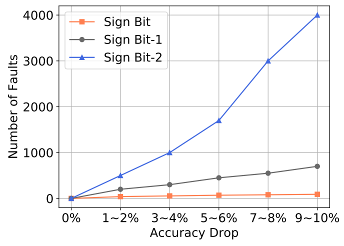
Table I: Overview of the MLP Architecture.

Dataset	Model Configuration
MLP on MNIST	$784 - 256 - 256 - 256 - 10$
MLP on Fashion-MNIST	$784 - 256 - 256 - 256 - 10$

1) Impact of Faults for Varying Bit Positions: In this experiment, the vulnerability of the network is analyzed for varying bit positions of the induced fault in the 8-bit weight stored in the DRAM. Bit-flips are introduced at random positions and



(a)



(b)

Figure 2: Impact of Faulty Bit Position on Classification Accuracy of MLP on (a) MNIST and (b) Fashion-MNIST.

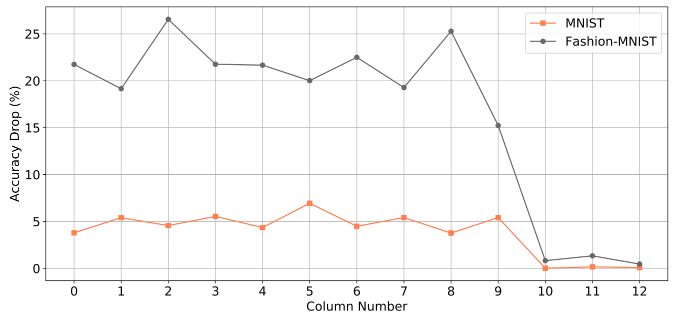


Figure 3: Variation of Accuracy Drop for 20 Faults across Varying Columns of the Weight Matrix.

Table II: Multiplication and additions in CNNs to classify one image [26]

CNN Architecture	Conv2d Layers	Linear Layers	Number of Multiplications	Number of Additions
LeNet-5	3	2	416,520	416,520
AlexNet	5	3	714,188,480	714,188,480
VGG-16	13	3	15,470,264,320	15,470,264,320
ResNet-50	53	1	3,729,522,688	1,761,820,672

the average classification accuracy of the model is observed for 10 runs. Bit flip faults are induced in the three most significant bit positions, starting from the sign bit.

The corresponding accuracy drop for MLPs on MNIST and Fashion-MNIST are represented in Figures 2a and 2b respectively. As seen from the figures, with the increase in number of faults, the classification accuracy of the network reduces for every bit position. As the significance of the bit position diminishes, the sensitivity of the induced fault reduces; thereby increasing the number of faults to accomplish identical reduction in classification accuracy. Since bit-flips induced in the sign bit reverses the signed integer representations, it causes maximum impact on the classification accuracy of the network. Hence, MLPs on MNIST and Fashion-MNIST manifest an 1-2% reduction in accuracy with only 250 and 40 faults injected in each layer, respectively, at the sign bit of the weights. Thus, minimal faults in the sign bit have the most intense impact on the reliability of the DNN accelerator.

2) Impact of Faults on the Most Vulnerable Weights: Random errors throughout the fault space furnish adequate degradation in classification accuracy. However, the number of random faults required to bring about such degradation is usually quite large. Hence, in this experiment, we focus on estimating the most vulnerable weights in a MLP. The weights corresponding to a particular layer in the MLP are arranged in the form of rows and columns in the 2D DRAM structure, where a particular column of weights corresponds to a specific neuron in the layer [4]. Hence, inducing faults along a distinct column is likely to produce a deeper impact on the network performance. As described in Section I-C1, since flipping the sign bit has the most impact on the model performance, we introduce bit-flip errors in the sign bit across only 20 random locations along a particular column of the weight matrix. This erroneous column location is varied, and the corresponding degradation in accuracy, averaged over 10 random runs, is depicted in Figure 3 for both the datasets. We observe that faults in all the columns till column ‘9’ render almost a consistent reduction in accuracy for all the datasets. However, as the column number exceeds ‘9’, the accuracy drop plummets close to zero, signifying almost negligible impact of the faults on the accuracy of the network. Since the output layer of the network consists of 10 neurons corresponding to 10 distinct classes in all the datasets, the last layer weights are mapped onto the first 10 columns of the matrix. When faults are induced in one among those 10 columns, the computation for that column corresponding to a specific neuron, and thereby to a particular application class is disrupted. With datasets having higher number of classes, such faults can impact differently, affecting the computation in other columns of the systolic array.

Therefore, by injecting bit-wise faults engendering from device-level non-idealities in the primary memory subsystem, we analyzed the reliability of resource-constrained DNN accelerators. We perform an extensive fault characterization of a neural network architecture on multivariate exhaustive datasets. An application-level analysis on the quantized pre-trained inference networks demonstrate degradation of classification accuracy, even at infinitesimal error rates. Hence, it is highly imperative to develop mitigation strategies that protect the most vulnerable network parameters in the memory, in order to improve the performance of the DNN accelerator at sub-optimal DRAM refresh rates.

II. AI/Deep Learning Accelerator Faults and Performance Impact

AI and Deep Learning tasks are compute-intensive and require millions of Multiply and Accumulate (MAC) operations for training and inference. In Table II, the number of MACs required for classifying a single image from ImageNet benchmark [25] during inference is shown. To increase the execution throughput, the AI accelerators are designed with thousands of densely packed Processing Elements(PEs)/MAC circuits [1]. As a result of this dense integration at advanced technology nodes which are susceptible to defect and yield issues, the AI chips are especially vulnerable to circuit faults (Fig. 4).

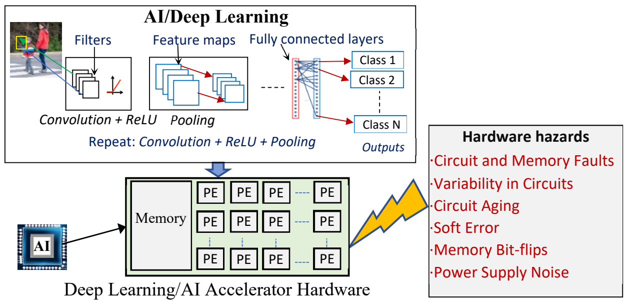


Figure 4: Hardware hazards and faults impacting AI accelerator chips.

A. Circuit and Hardware Hazards on Deep Learning/AI Accelerator Fidelity

The major types of circuit and transistor level hazards that can impact the performance of AI/Deep L
are, (i) process variation induced circuit pa
(ii) runtime power supply voltage noise a
cuit aging with time, and (iv) radiation i
For safety-critical applications, chips with
faults are generally discarded according to
million/billion (DPPM/DPPB) guidelines
[27]. However, aging induced in-field stuck-a
concern for automotives.

1) Process Variations:: Process parameter variations are caused by semiconductor manufacturing imperfections, which is a major concern for current advanced technologies (e.g., 10nm and newer), and impact circuit performance at both the memory and MAC units in accelerator hardware. As a result, different samples of the same accelerator chip might exhibit different frequencies (i.e., speed binning [28]) post-fabrication. For safety-critical AV application, it is important to analyze how process variations can impact the Deep Learning accuracy and fidelity. Unlike other areas, a one-size-fits-all training method may not be suitable for Deep Learning used in safety-critical domains.

2) Power Supply Noise:: Voltage noise is caused by simultaneous switching events inside the chip. Since the accelerator chips will perform millions of operations to infer decisions from analysis of the images, the extensive switching inside the MAC and memory units from this can cause voltage noise [29]. The corresponding transient voltage droop can cause timing violations or bit-flips inside the accelerator.

3) Circuit Aging:: Circuit aging is a critical reliability problem in modern VLSI chips [30]. Since aging is use-case dependent and cannot be accurately estimated at time 0, it is a major concern for safety-critical applications. Aging can impact the weight storage SRAM modules by altering their read/write stability with time, and thus cause bit-flip errors. The Deep Learning accelerator's performance (i.e., operating frequency F_{MAX}) might shift with time depending on the amount of usage.

4) Soft Errors:: Although most of the AI accelerators are used at sea-level altitude, it may seem they are immune to soft errors caused by high-energy neutrons from cosmic radiation. However, as shown in detail in [31], because of weight reuse in Deep Learning, soft-errors can indeed impact the accuracy of accelerators. Hence, it is essential to embed resiliency and robustness against random bit-flips in addition to radiation hardening of the critical MSB bits.

The key modules of accelerators that are susceptible to defects are the weight storage SRAM/Register Files (RF) and the PE/MACs. For SRAM/RF, the faults are generally repaired with ECC and spare cells. The timing-faults in MAC can be solved by appropriate timing guard-band and run-time frequency adjustment. The stuck-at faults in the MAC are permanent and can adversely impact accuracy. In this section accuracy impact of stuck-at faults in MAC are analyzed.

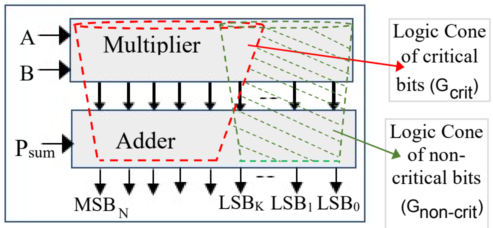

A
B
Multiplier
Psum
Adder
MSB N
LSBK LSB1 LSB0
Logic Cone of critical bits (Gcrit)
Logic Cone of non-critical bits (Gnon-crit)

Figure 5: Faults are classified as critical or non-critical based on their location in the logic cone of the bits.

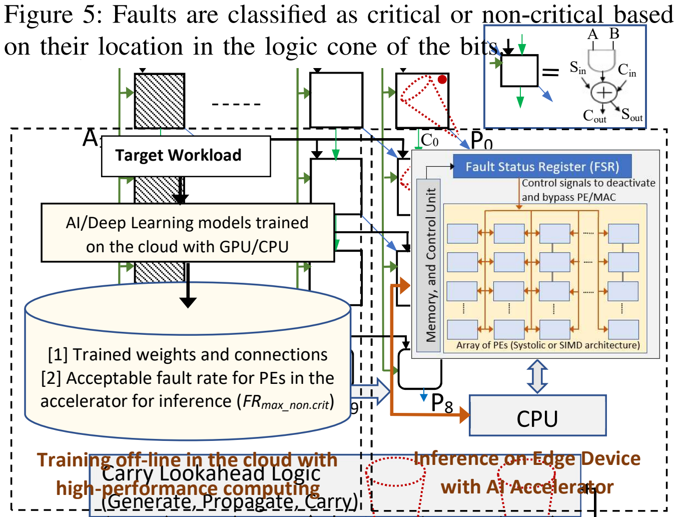
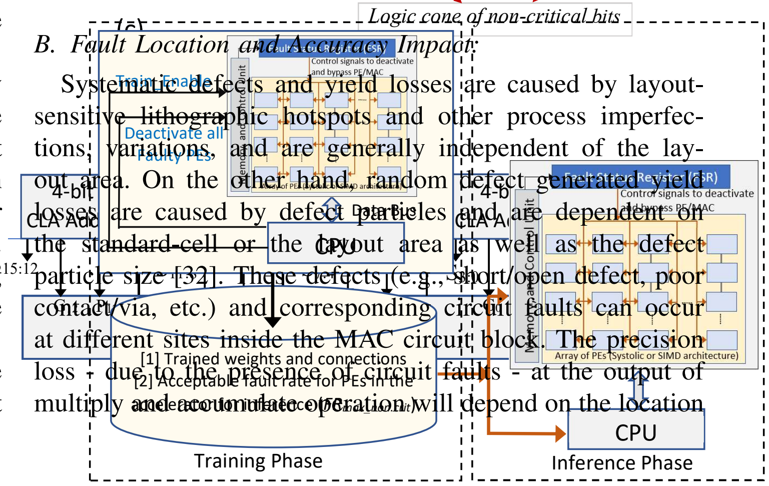

Target Workload
AI/Deep Learning models trained on the cloud with GPU/CPU
[1] Trained weights and connections
[2] Acceptable fault rate for PEs in the accelerator for inference (FRmax_non.crit)
Fault Status Register (FSR)
Control signals to deactivate and bypass PE/MAC
Memory, and Control Unit
Array of PEs (Systolic or SIMD architecture)
CPU
Training off-line in the cloud with high-performance computing
Inference on Edge Device with AI Accelerator

Figure 6: Deactivating some non-critically faulty PEs to keep fault rate within $FR_{max\_non-crit}$ to ensure accuracy does not degrade beyond acceptable limit.

B. Fault Location and Accuracy Impact::

Systematic defects and yield losses are caused by layout-sensitive lithographic hotspots and other process imperfections, variations, and are generally independent of the layout area. On the other hand, random defect generated yield losses are caused by defect particles and are dependent on the standard-cell or the layout area as well as the defect particle size [32]. These defects (e.g., short/open defect, poor contact/via, etc.) and corresponding circuit faults can occur at different sites inside the MAC circuit block. The precision loss - due to the presence of circuit faults - at the output of multiply and accumulate operation will depend on the location


Deactivate all faulty PEs
Fault Status Register (FSR)
Control signals to deactivate and bypass PE/MAC
Array of PEs (Systolic or SIMD architecture)
CPU
Training Phase
Inference Phase

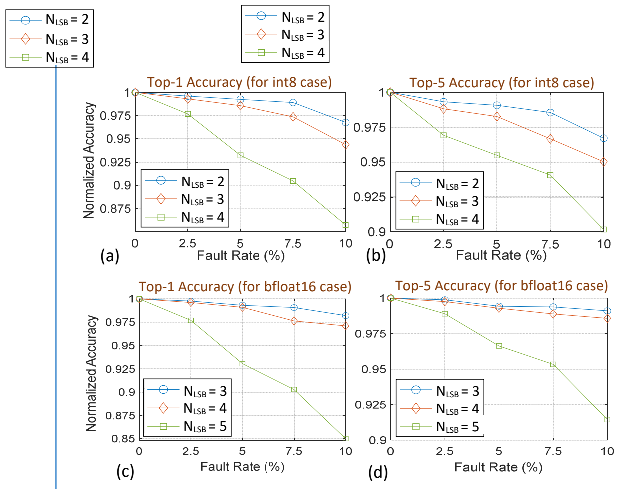
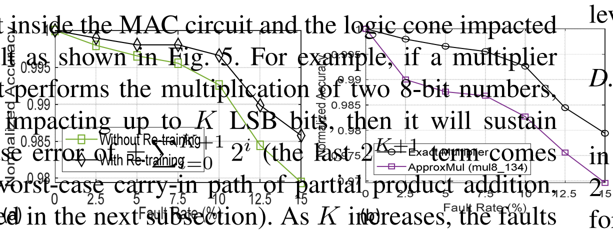


Figure 7: Impact of faults in logic cones of LSB positions on inference accuracy for AlexNet with ImageNet data set. (a), (b) int8 data format and MAC type.; (c), (d) bfloat16 data format and MAC type.



of the fault inside the MAC circuit and the logic cone impacted by the fault as shown in Fig. 5. For example, if a multiplier circuit that performs the multiplication of two 8-bit numbers, has faults impacting up to K LSB bits, then it will sustain a worst-case error of $\pm \sum_{i=0}^{K+1} 2^i$ (the last 2^{K+1} term comes from the worst-case carry-in path of partial product addition, as explained in the next subsection). As K increases, the faults impact the more significant digits causing the worst-case error of the multiplier to increase. Errors will also occur in the adder circuit of the MAC if it is corrupted by faults. Since the multiplier is the more area-dominant block in a MAC, it will be more prone to faults and computation errors. A similar analysis can be also done for bfloat16 [33] MAC to identify faults in LSB positions (i.e., logic cone of corresponding bits) and their impact on accuracy. If any PE has a fault in the critical logic cone then it needs to be deactivated to prevent accuracy loss because critical faults can cause large magnitude error in MAC output. For faults in the non-critical logic cone, up to a certain fault rate (i.e., $FR_{max\_non-crit}$) may be acceptable.

The IDs of faulty PEs and their fault type (e.g., critical or non-critical) can be identified with ATPG test patterns and recorded in a Fault Status Register (FSR) [26]. During inference, the mobile/edge accelerator's FSR and control unit reads the $FR_{max\_non-crit}$ and if it is lower than the current fault rate $FR_{non-crit}$, then the control unit sends deactivation signals to disable some of the PEs. The deactivation protocol and the complete map of faulty PE locations are programmed in firmware/software by the manufacturer. With the user-given input of acceptable fault rate and the stored PE fault map, the protocol will automatically disable a few faulty PEs to ensure that the overall faulty PE rate of the accelerator does not exceed a threshold (i.e., $FR_{max\_non-crit}$). When deactivating some faulty PEs, the firmware/software will ensure that the remaining faulty PEs are not clustered. As shown in Fig. 6, control signals are transmitted from the FSR to all the PEs to selectively disable the faulty PEs when needed. By adopting this scheme, the manufacturer can avoid discarding the full accelerator chip/die only because of the presence of few PEs with faulty MACs, and thereby increase yield. The overhead in this yield loss reduction are the extra on-chip register (FSR) to store the IDs and the control signal routes to disable faulty PEs [26].

C. Accelerator Architecture and Faulty PE Deactivation

AI accelerators can be designed either in SIMD or Systolic architectures. In SIMD architecture the loosely coupled independent Processing Elements (PEs) are connected with NoC or mesh and can be individually switched off and bypassed with wires [1], [2], [26]. The tightly-coupled 2D systolic-arrays in TPU introduce challenges in deactivating and bypassing individual PEs. A common technique is to use spare PE blocks to substitute for faulty PEs. However, this approach requires complex wiring between spare and faulty PEs [34]. An innovative software-level technique to deactivate and bypass faulty PEs in systolic array without any hardware-level modifications was proposed in [26].

D. Results and Analysis

To identify the impact of the number of LSBs having faults in their logic cones, the number of LSBs were varied from 2 to 4 for int8 and 3 to 5 for bfloat16 [33] data formats for the CNN model AlexNet with the ImageNet test set. These experiments were done with PyTorch. The results are shown in Fig. 7, where X-axis is fault-rate in non-critical logic cones (the LSBs). From this analysis, we conservatively selected 2 LSBs for int8 and 4 mantissa LSBs for the bfloat16 hardware models as non-critical (i.e., circuit faults occurring in their logic cones are non-critical), and rest of all the bits are considered critical. Any PE with one or more critical faults are considered defective and must be deactivated to ensure fidelity in accuracy. In the experiments the accelerator hardware comprised of 128 by 128 array of PE/MACs. The faulty PEs are modeled as uniformly distributed across the accelerator columns with fault probability of $FR\%$, the fault rate. This implies that for $FR\%$ fault rate, each column of the accelerator has $0.01 * FR * N_{Row}$ faulty PEs randomly distributed across that column.

The accuracy changes - in the standard Top-1 and Top-5 format - with MAC faults are shown in Fig. 8 for 50,000 test images from ImageNet [25]. We observe from Fig. 8 that up to a certain fault (non-critical) rate might be acceptable depending on the desired accuracy of the AI/Deep Learning task.

1) Fault-aware training to enhance Robustness: Using a fault-aware training flow some of the accuracy loss due to faults in MAC units can be recovered by incorporating the fault effects in the backpropagation-based weight update segment and allowing the DNN to adapt accordingly [26]. To experimentally demonstrate this technique, we used the LeNet-5 CNN architecture. Results from this fault-aware training are shown in Fig. 9. For 7.5% fault rate (non-critical faults) the

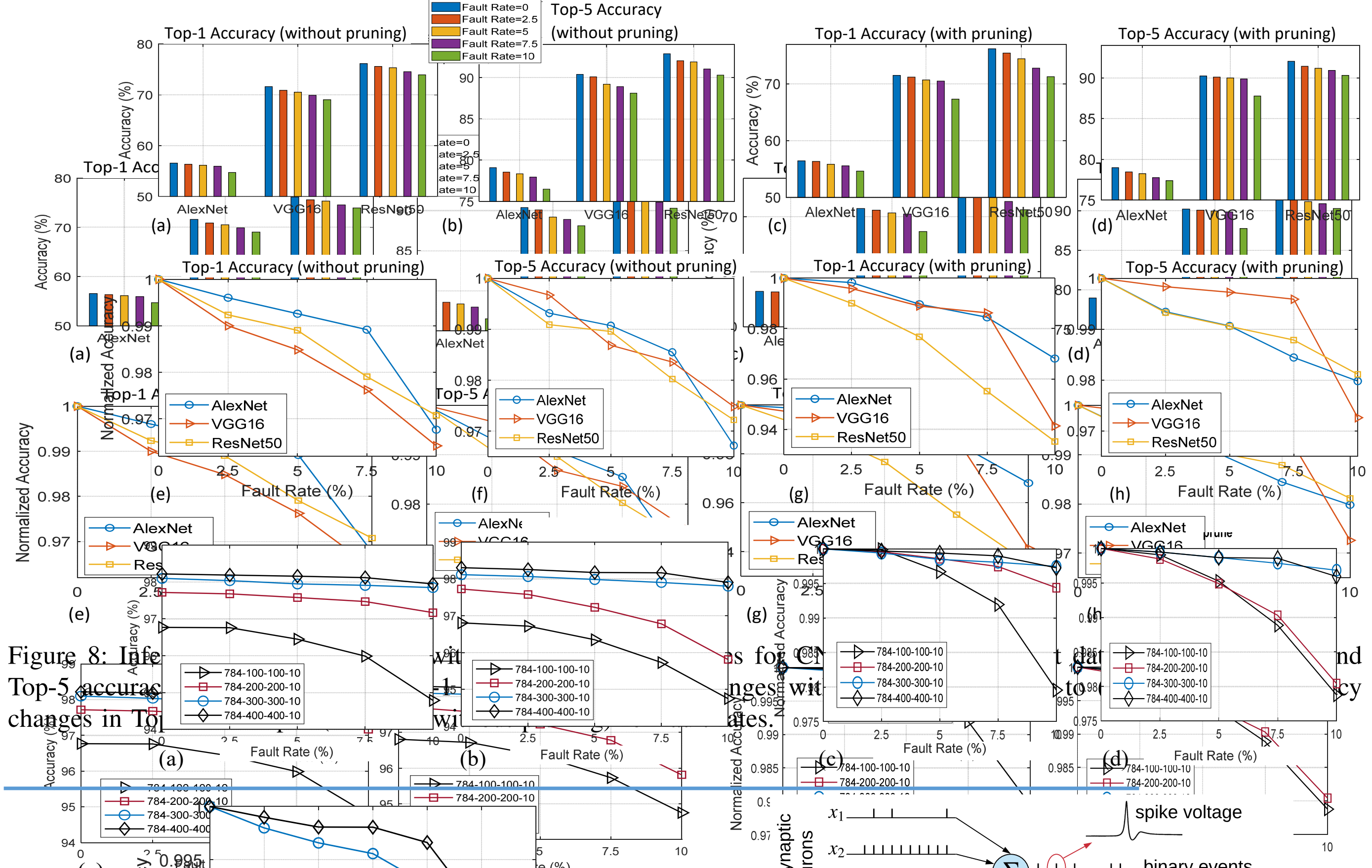


Figure 8: Infe... wit... s for C... gat... nd Top-5 accurac... -1 ... nges wit... to... cy changes in To... wi... ates.

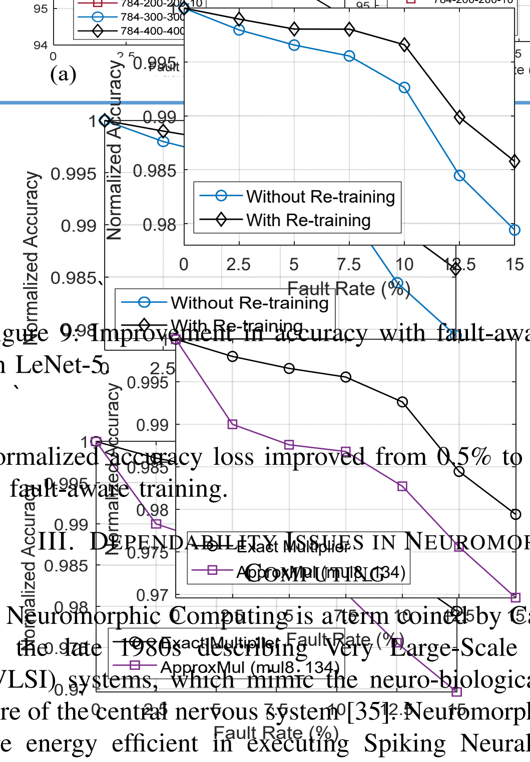


Figure 9: Improvement in accuracy with fault-aware training on LeNet-5.

normalized accuracy loss improved from 0.5% to 0.22% due to fault-aware training.

III. Dependability Issues in Neuromorphic Computing

Neuromorphic Computing is a term coined by Carver Mead in the late 1980s describing Very Large-Scale Integration (VLSI) systems, which mimic the neuro-biological architecture of the central nervous system [35]. Neuromorphic systems are energy efficient in executing Spiking Neural Networks (SNNs), which are considered as the third generation of neural networks. SNNs use spike-based computations and bio-inspired learning algorithms in solving pattern recognition problems [36]. In an SNN, pre-synaptic neurons communicate information encoded in spike trains to post-synaptic neurons, via synapses. Performance, e.g., accuracy of an SNN model can be assessed in terms of the inter-spike interval (ISI), which is defined as inverse of the mean firing rate of neurons. This is illustrated in Figure 10.

Recently, neuromorphic platforms such as DYNAPs [37], TrueNorth [38], and Loihi [39] are introduced to the systems

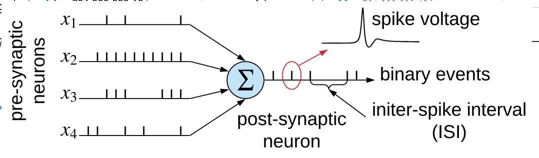


Figure 10: Integration of spike trains at the post-synaptic neuron from four pre-synaptic neurons in a Spiking Neural Network (SNN). Each spike is a voltage waveform of small time duration to the order of ms.

community. These systems are designed as tile-based architectures with a shared interconnect for communication [40]–[42]. A tile may consist of a crossbar for mapping neurons and synapses of an application. Recently, Non-Volatile Memory (NVM) technologies such as Phase-Change Memory (PCM) and Oxide-based Resistive RAM (OxRRAM) are used to implement synaptic storage in each crossbar [43], [44].* NVMs bring certain advantages such as high integration density, multi-bit synapses, CMOS compatibility, and above all, non-volatility, which can further lower the energy consumption of neuromorphic hardware. However, NVMs also introduce reliability issues such as endurance, aging, and read disturbances (see Table III for a summary of these issues). In this paper, we will discuss some of these issues and our approach to mitigate them.

A. Introduction to Non-Volatile Memory

Emerging NVM technologies such as phase-change memory (PCM), oxide-based memory (OxRAM), spin-based magnetic memory (STT-MRAM), and Flash have recently been used

*Beside neuromorphic computing, some of these NVM technologies are also used to implement main memory in conventional computers to improve performance and energy efficiency [45]–[49].

Table III: Reliability issues in NVMs.

Reliability Issues	NVMs
High-voltage related circuit aging	PCM, Flash
High-current related circuit aging	OxRAM, STT-MRAM
Read disturbance	All
Limited endurance	All

to implement synaptic storage in neuromorphic hardware. NVMs are non-volatile, have high CMOS compatibility, and can achieve high integration density. Each NVM device can implement both a single-bit and multi-bit synapse. Because of these properties, an NVM-based neuromorphic hardware typically consumes energy that is in the order of magnitudes lower than using SRAMs [43].

Without loss of generality, we discuss the dependability issues for PCM-based neuromorphic computing. PCM is one of the mature NVM technologies and is successfully demonstrated in recent neuromorphic prototypes [44].

Figure 11 ❶ illustrates how a chalcogenide semiconductor alloy is used to build a PCM cell. The amorphous phase (RESET) in this alloy has higher resistance than the crystalline phase (SET). $Ge_2Sb_2Te_5$ (GST) is the most commonly used alloy for PCM. To compute $(x_i \cdot w_i)$, a current is injected into the resistor-chalcogenide junction via the heater element. The current is controlled to ensure that the phase of the PCM cell is not disturbed. This is the fundamental operation of forward propagation of neuron excitation during inference. For online learning (e.g., using STDP), the injected current induces (Joule) heating in the chalcogenide alloy, changing its conductivity, thereby achieving synaptic weight updates. Figure 11 ❷ illustrates the current profiles necessary for inference (using the read pulse) and online learning (using SET and RESET pulses) in PCM. These current profiles are generated using an on-chip charge pump (CP). Figure 11 ❸ illustrates the PCM cell's operation when idle, i.e., when a neuron is not activated. We illustrate a 1D-1R structure, where a single PCM cell is connected to a row and column using a diode as an access device. Diode-based PCM cells allow very high integration density in scaled technology nodes compared to transistor-based PCM. The CP is operated at 1.8V to maintain the required biasing. Finally, Figure 11 ❹ illustrates the PCM operation during inference. The CP is operated at 3V to generate the read current profile of Figure 11 ❷ using the sense amplifier (SA). The write driver (WD) is used for generating the currents for online learning.

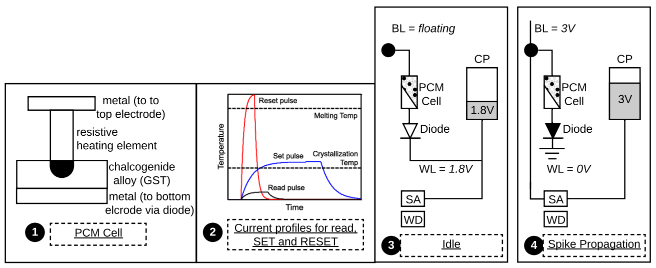


Figure 11: Operation of PCM in neuromorphic computing.

These high-voltage operations of the charge pump (and the peripheral circuit of a crossbar) accelerate circuit aging, lowering the dependability of neuromorphic computing.

Apart from diode, transistors are also used as access devices. When using transistor-based PCM cells, the CP is operated at lower voltages: 1.2V during idle and 1.8V during spike propagation. Though aging issues are less severe in such designs, they are still a dependability concern for neuromorphic computing.

Peripheral aging are not the only dependability issues in neuromorphic hardware. Unfortunately, NVMs have limited endurance, ranging from 10^5 (for Flash) to 10^{10} (for OxRRAM), with PCM somewhere in between ($\approx 10^7$).

In the following we describe our recent efforts in mitigating aging and endurance in neuromorphic computing.

B. Mitigating Peripheral Aging in Neuromorphic Hardware

In our prior work [50]–[52], we have analyzed different aging issues in the peripheral circuit of a neuromorphic hardware. We briefly elaborate on these issues.

We consider the CMOS aging due to Time-Dependent Dielectric Breakdown (TDDB), Bias Temperature Instability (BTI), and Hot-Carrier Injection (HCI) failure mechanisms. These are the dominant ones in scaled technology nodes (45nm and below). In older nodes, Electromigration (EM) also plays a key role [53]–[70].

CMOS aging is accelerated when the device is *stressed*, i.e., exposed to high overdrive voltages[*]. With this understanding, we provide a brief background of these failure mechanisms.

- *TDDB:* This is a failure mechanism in a CMOS device, when the gate oxide breaks down as a result of long-time application of relatively low electric field (as opposed to immediate breakdown, which is caused by strong electric field) [71]. The lifetime of a CMOS device is measured in terms of its *mean time to failure* (**MTTF**) as
$$\text{MTTF}_{\text{TDDB}} = A.e^{-\gamma\sqrt{V}}, \tag{1}$$
where A and γ are material-related constants, and V is the overdrive gate voltage of the CMOS device.
- *BTI:* This is a failure mechanism in a CMOS device in which positive charges are trapped at the oxide-semiconductor boundary underneath the gate [72]. BTI manifests as 1) decrease in drain current and transconductance, and 2) increase in off current and threshold voltage. The BTI lifetime of a CMOS device is
$$\text{MTTF}_{\text{BTI}} = \frac{A}{V^{\gamma}} e^{\frac{E_a}{KT}}, \tag{2}$$
where A and γ are material-related constants, E_a is the activation energy, K is the Boltzmann constant, T is the temperature, and V is the overdrive gate voltage.
- *HCI:* This is a failure mechanism in a CMOS device, when a carrier (electron or hole) gains sufficient kinetic energy to overcome the potential barrier of the conducting channel and gets trapped in the gate dielectric, permanently changing the CMOS's switching properties [73].

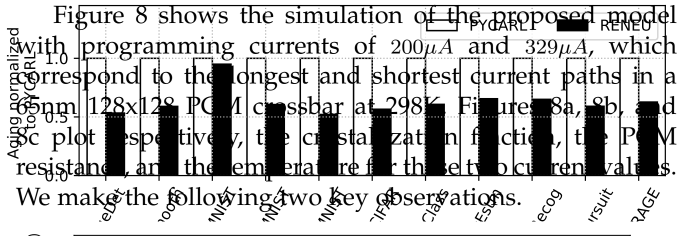
[*]Overdrive voltage is defined as the voltage between transistor gate and source (V_{GS}) in excess of the threshold voltage (V_{th}), where V_{th} is the minimum voltage required between gate and source to turn the transistor on.

Unlike the TDDB and BTI failure mechanisms, for which silicon-characterized reliability models are available from foundries, characterized models for HCI failure mechanism are still in development for scaled nodes.

Current methods for qualifying reliability are overly conservative, since they estimate circuit aging considering worst-case operating conditions and unnecessarily constrain performance. Recent system-level works on mapping SNN-based applications to neuromorphic hardware, such as [74]–[83], target performance improvement only. They do not consider reliability issues of neuromorphic computing.

To address these limitations, we have designed RENEU [51], a reliability-oriented approach to map machine learning applications to neuromorphic hardware, with the aim of improving system-wide reliability, without compromising key performance metrics such as execution time of these applications on the hardware. Fundamental to RENEU is a novel formulation of the aging of CMOS-based circuits in a neuromorphic hardware considering different failure mechanisms. Using this formulation, RENEU develops a system-wide reliability model which can be used inside a design-space exploration framework involving the mapping of neurons and synapses to the hardware. To this end, RENEU uses an instance of Binary Particle Swarm Optimization (PSO) [84] to generate mappings that are Pareto-optimal in terms of performance and reliability. We evaluate RENEU using SNN-based streaming and non-streaming applications [85]–[89]. We evaluate these applications on a state-of-the-art neuromorphic hardware with PCM synapses.

Figure 12 reports the circuit aging caused by RENEU normalized to PyCARL, a state-of-the-art SNN mapping approach. We plot results for each of our machine learning



F
n

C. Mitigating Thermal and Endurance Issues in Neuromor-
p

is
e

We analyze the internal architecture of a PCM crossbar (see F

and vertical wires of a crossbar are a major source of parasitic voltage drops in the crossbar. Using detailed circuit simulations at different process (P), voltage (V), and temperature (T) corners, we show that these voltage drops create current variations in the crossbar. For the same spike voltage, current on the shortest path is significantly higher than the current on the longest path in the crossbar, where the length of a current path is measured in terms of its number of parasitic components. These current variations create asymmetry in the self-l
upda
learn

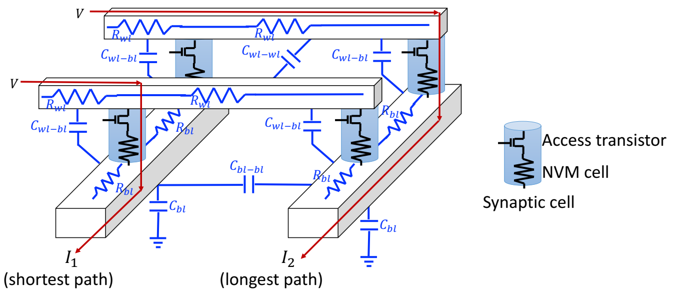
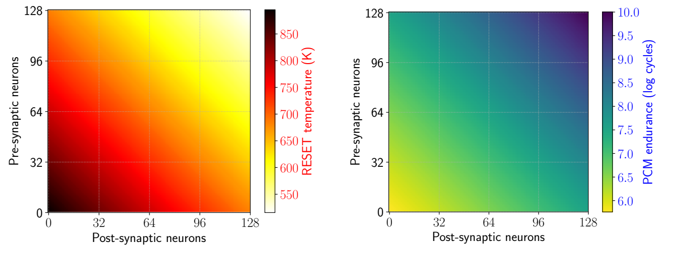


Figure 13: Bitline and wordline parasitics in a PCM crossbar.

Figure 14 plots the temperature and endurance maps of a 128x128 crossbar at 65nm process node with $T_{amb} = 298K$. The PCM cells at the bottom-left corner have higher self-heating temperature than at the top-right corner. This asymmetry in the self-heating temperature creates a wide distribution of endurance, ranging from 10^6 cycles for PCM cells at the bottom-left corner to 10^{10} cycles at the top-right corner. These endurance values are consistent with the values reported for recent PCM chips from IBM [94].



(a) Thermal map for PCM RESET operations in a 128x128 crossbar.

(b) Endurance map of the PCM cells in a 128x128 crossbar.

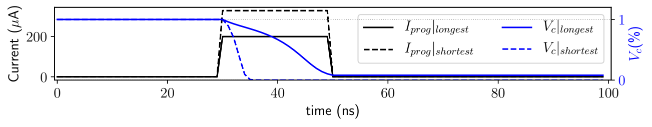
Figure 14: Temperature and endurance map of a 128x128 PCM crossbar at 65nm process node with $T_{amb} = 298K$.

Using such technology modeling, we propose eSpine, a novel technique to improve endurance-related lifetime of PCM by incorporating the endurance variation within each crossbar in mapping machine learning workloads, ensuring that synapses with higher activation are always implemented on PCM cells with higher endurance, and vice versa. eSpine works in two steps. First, it uses the Kernighan-Lin Graph Partitioning algorithm to partition a workload into clusters of neurons and synapses, where each cluster can fit in a crossbar. Second, it uses an instance of PSO to map clusters to tiles, where the placement of synapses of a cluster to the PCM cells of a crossbar is performed by analyzing their activation within a workload. We evaluate eSpine for a state-of-the-art

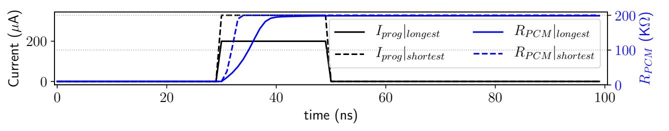
4.1 Model Prediction

The thermal and endurance models in Equations 6 and 10, respectively are integrated as follows. The self-heating temperature of Equation 6 is first computed using the PCM's programming current. This self-heating temperature is then used to compute the endurance using Equation 10.

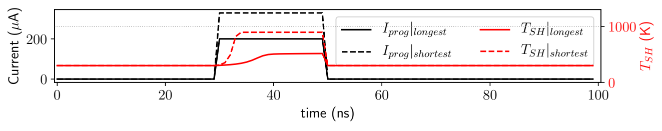
Figure 8 shows the simulation of the proposed model with programming currents of $200\mu A$ and $329\mu A$, which correspond to the longest and shortest current paths in a 65nm 128x128 PCM crossbar at 298K. Figures 8a, 8b, and 8c plot respectively, the crystallization fraction, the PCM resistance, and the temperature for these two current values. We make the following two key observations.



(a) Change in crystallization fraction of PCM.



(b) Change in PCM resistance.



(c) Change in PCM temperature.

Fig. 8. Validation of the proposed model.

Fig. 9. Temperature and endurance map of a 128x128 crossbar at 65nm process node with $T_{amb} = 298K$.

the memristor connecting the i^{th} pre-synaptic neuron with j^{th} post-synaptic neuron in a memristive crossbar as

$$\mathcal{L}_{i,j} = \mathcal{E}_{i,j}/a_{i,j} \quad (11)$$

where $a_{i,j}$ is the number of synaptic activations of the memristor in a given SNN workload and $\mathcal{E}_{i,j}$ is its endurance. Equation 11 combines the effect of software (SNN mapping) on hardware (endurance and temperature). eSpine aims to maximize the minimum normalized lifetime, i.e.,

$$\mathcal{L}_{sys} = \text{maximize}\{\min_{i,j} \mathcal{L}_{i,j}\} \quad (12)$$

In most earlier works on wear-leveling in the context of non-volatile main memory (e.g., Flash), lifetime is computed in terms of utilization of NVM cells, ignoring the variability of endurance within the device. Instead, we formulate the effective lifetime by considering a memristor's endurance and its utilization in a workload. This is to allow cells with higher endurance to have higher utilization in a workload.

neuromorphic hardware model with PCM synapses. Using 10 SNN workloads, we demonstrate a significant improvement in the effective lifetime.

IV. Conclusion

To conclude, as process technology continues to scale aggressively, reliability issues in neuromorphic hardware are becoming a primary concern for system developer. First, we analyzed the reliability of DNN accelerators in the presence of DRAM faults. By injecting bit-wise faults engendering from device-level non-idealities in the memory subsystem of the accelerator, we perform an extensive fault characterization of multiple DNN architectures on multivariate exhaustive datasets. An application-level analysis on the quantized pre-trained inference networks demonstrate degradation of classification accuracy, even at infinitesimal error rates. Next, we analyzed the MAC circuits, as they occupy a significant portion of the DNN hardware. Circuit faults in MSB logic cones of the MAC can adversely impact accuracy. For MACs with faults in LSB logic cones, the fraction of such faulty MACs must be within an application-dependent range to ensure accuracy degradation does not exceed acceptable limits. System-level approaches, such as the ones we highlighted in this paper, mitigate these reliability issues via intelligent neuron and synapses placement on the hardware. These approaches, however, can be further improved by incorporating technology perspective within a holistic design-technology co-optimization framework.

References

[1] Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*, 2017, pp. 1–12.

[2] Chen *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, pp. 127–138, 2016.

[3] Park *et al.*, "4.6 a1. 93tops/w scalable deep learning/inference processor with tetra-parallel mimd architecture for big-data applications," in *IEEE ISSCC*, 2015, pp. 1–3.

[4] Zhang *et al.*, "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.

[5] Liu *et al.*, "Flikker: saving dram refresh-power through critical data partitioning," in *Proceedings of the 16th ASPLOS*, 2011, pp. 213–224.

[6] Lucas *et al.*, "Sparkk: Quality-scalable approximate storage in dram," in *The memory forum*, 2014, pp. 1–9.

[7] Sampson *et al.*, "Enerj: Approximate data types for safe and general low-power computation," *ACM SIGPLAN Notices*, vol. 46, pp. 164–174, 2011.

[8] Rahmati *et al.*, "Refreshing thoughts on dram: Power saving vs. data integrity," in *Workshop on Approximate Computing Across the System Stack*, 2014.

[9] Raha *et al.*, "Synergistic approximation of computation and memory subsystems for error-resilient applications," *IEEE Embedded Systems Letters*, vol. 9, pp. 21–24, 2017.

[10] Raha *et al.*, "Quality configurable approximate dram," *IEEE TC*, vol. 66, pp. 1172–1187, 2016.

[11] S. Cass, "Nvidia makes it easy to embed ai: The jetson nano packs a lot of machine-learning power into diy projects-[hands on]," *IEEE Spectrum*, vol. 57, pp. 14–16, 2020.

[12] S. Cass, "Taking ai to the edge: Google's tpu now comes in a maker-friendly package," *IEEE Spectrum*, vol. 56, pp. 16–17, 2019.

[13] "Intel movidius™ myriad™ x vision processing unit (vpu)," https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu/movidius-myriad-x.html, (Accessed on 03/17/2021).

[14] Chen *et al.*, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE JETCAS*, vol. 9, pp. 292–308, 2019.

[15] "Nvidia ampere architecture — nvidia," https://www.nvidia.com/en-us/data-center/nvidia-ampere-gpu-architecture/, (Accessed on 08/01/2020).

[16] "Intel keem bay vpu," https://d1io3yog0oux5.cloudfront.net/_a000324c17af610f0c3d7c52e5620775/intel/db/861/7791/pdf/intel-ai-summit-keynote-slides.pdf, (Accessed on 10/02/2020).

[17] Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *ACM Design Automation Conference (DAC)*, 2018, pp. 1–6.

[18] Schorn *et al.*, "An efficient bit-flip resilience optimization method for deep neural networks," in *Design, Automation & Test in Europe Conference & Exhibition*, 2019, pp. 1507–1512.

[19] Jha *et al.*, "Ml-based fault injection for autonomous vehicles: a case for bayesian fault injection," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 112–124.

[20] Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[21] Chen *et al.*, "Binfi: an efficient fault injector for safety-critical machine learning systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–23.

[22] Kundu *et al.*, "High-level modeling of manufacturing faults in deep neural network accelerators," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2020, pp. 1–4.

[23] Zhang *et al.*, "Fault-tolerant systolic array based accelerators for deep neural network execution," *IEEE Design & Test*, vol. 36, pp. 44–53, 2019.

[24] Kundu *et al.*, "Toward functional safety of systolic array-based deep learning hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, pp. 485–498, 2021.

[25] J. D. et al., "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[26] M. Sadi and U. Guin, "Test and yield loss reduction of ai and deep learning accelerators," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[27] ISO 26262, "Road vehicles-Functional safety," 2018, Accessed: 11-Mar-2021.

[28] M. Sadi et al., "SoC Speed Binning Using Machine Learning and On-Chip Slack Sensors," in *IEEE Transactions on Computer-Aided Design*, 2017.

[29] M. Sadi and M. Tehranipoor, "Design of a Network of Digital Sensor Macros for Extracting Power Supply Noise Profile in SoCs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2016.

[30] M. Sadi, G. K. Contreras, J. Chen, L. Winemberg, and M. Tehranipoor, "Design of Reliable SoCs With BIST Hardware and Machine Learning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017.

[31] G. Li et al., "Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications," in *International Conference for High Performance Computing, Storage and Analysis*, 2017.

[32] A. J. Strojwas et al., "Yield and Reliability Challenges at 7nm and Below," in *Electron Devices Technology and Manufacturing Conference*, 2019.

[33] S. Wang and P. Kanwar, "BFloat16: The secret to high performance on Cloud TPUs," in *Google Cloud Blog*, 2019.

[34] J. J. Zhang, K. Basu, and S. Garg, "Fault-Tolerant Systolic Array Based Accelerators for Deep Neural Network Execution," in *IEEE Design & Test*, 2019.

[35] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, 1990.

[36] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, 1997.

[37] S. Moradi, N. Qiao, F. Stefanini *et al.*, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *TBCAS*, 2017.

[38] M. V. Debole, B. Taba, A. Amir *et al.*, "TrueNorth: Accelerating from zero to 64 million neurons in 10 years," *Computer*, 2019.

[39] M. Davies, N. Srinivasa, T. H. Lin *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, 2018.

[40] A. Balaji, Y. Wu, A. Das, F. Catthoor, and S. Schaafsma, "Exploration of segmented bus as scalable global interconnect for neuromorphic computing," in *GLSVLSI*, 2019.

[41] A. Balaji, S. Ullah *et al.*, "Design methodology for embedded approximate artificial neural networks," in *GLSVLSI*, 2019.

[42] F. Catthoor, S. Mitra, A. Das, and S. Schaafsma, "Very large-scale neuromorphic systems for biological signal processing," in *CMOS Circuits for Biological Sensing and Processing*. Springer, 2018.

[43] A. Mallik, D. Garbin, A. Fantini, D. Rodopoulos, R. Degraeve *et al.*, "Design-technology co-optimization for OxRRAM-based synaptic processing unit," in *VLSIT*, 2017.

[44] G. W. Burr, R. M. Shelby *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, 2017.

[45] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Enabling and exploiting partition-level parallelism (PALP) in phase change memories," *TECS*, 2019.

[46] S. Song, A. Das, and N. Kandasamy, "Exploiting inter-and intra-memory asymmetries for data mapping in hybrid tiered-memories," in *ISMM*, 2020.

[47] S. Song, A. Das, O. Mutlu *et al.*, "Improving phase change memory performance with data content aware access," in *ISMM*, 2020.

[48] S. Song, A. Das, O. Mutlu *et al.*, "Aging aware request scheduling for non-volatile main memory," in *ASP-DAC*, 2021.

[49] S. Song and A. Das, "Design methodologies for reliable and energy-efficient PCM systems," in *IGSC Workshops*, 2020.

[50] A. Balaji, S. Song, A. Das, N. Dutt, J. Krichmar *et al.*, "A framework to explore workload-specific performance and lifetime trade-offs in neuromorphic computing," *CAL*, 2019.

[51] S. Song, A. Das *et al.*, "Improving dependability of neuromorphic computing with non-volatile memory," in *EDCC*, 2020.

[52] S. Song and A. Das, "A case for lifetime reliability-aware neuromorphic computing," in *MWSCAS*, 2020.

[53] A. Das, A. Kumar, and B. Veeravalli, "Reliability and energy-aware mapping and scheduling of multimedia applications on multiprocessor systems," *TPDS*, 2015.

[54] A. Das, A. Kumar, and B. Veeravalli, "Aging-aware hardware-software task partitioning for reliable reconfigurable multiprocessor systems," in *CASES*, 2013.

[55] A. Das, A. Kumar, B. Veeravalli, C. Bolchini, and A. Miele, "Combined dvfs and mapping exploration for lifetime and soft-error susceptibility improvement in mpsocs," in *DATE*, 2014.

[56] A. Das, A. Kumar, and B. Veeravalli, "Reliability-driven task mapping for lifetime extension of networks-on-chip based multiprocessor systems," in *DATE*, 2013.

[57] A. K. Das, A. Kumar, B. Veeravalli, and F. Catthoor, *Reliable and Energy Efficient Streaming Multiprocessor Systems*. Springer, 2018.

[58] A. Das, A. Kumar, and B. Veeravalli, "Fault-tolerant network interface for spatial division multiplexing based network-on-chip," in *ReCoSoC*, 2012.

[59] A. Das, G. V. Merrett, and B. M. Al-Hashimi, "The slowdown or race-to-idle question: Workload-aware energy optimization of smt multicore platforms under process variation," in *DATE*, 2016.

[60] A. Das, A. K. Singh, and A. Kumar, "Energy-aware dynamic reconfiguration of communication-centric applications for reliable mpsocs," in *ReCoSoC*, 2013.

[61] A. Das, G. V. Merrett, M. Tribastone, and B. M. Al-Hashimi, "Workload change point detection for runtime thermal management of embedded systems," *TCAD*, 2015.

[62] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar *et al.*, "Reinforcement learning-based inter-and intra-application thermal optimization for lifetime improvement of multicore systems," in *DAC*, 2014.

[63] A. Das, S. Venkataraman, and A. Kumar, "Improving autonomous soft-error tolerance of FPGA through LUT configuration bit manipulation," in *FPL*, 2013.

[64] R. Santos, S. Venkataraman, A. Das, and A. Kumar, "Criticality-aware scrubbing mechanism for sram-based fpgas," in *FPL*, 2014.

[65] A. Das, A. Kumar, and B. Veeravalli, "Communication and migration energy aware task mapping for reliable multiprocessor systems," *FGCS*, 2014.

[66] A. Das, A. Kumar, and B. Veeravalli, "Energy-aware task mapping and scheduling for reliable embedded computing systems," *TECS*, 2014.

[67] A. Das, A. Kumar, and B. Veeravalli, "Energy-aware communication and remapping of tasks for reliable multimedia multiprocessor systems," in *ICPADS*, 2012.

[68] A. Das, A. Kumar, and B. Veeravalli, "Temperature aware energy-reliability trade-offs for mapping of throughput-constrained applications on multimedia MPSoCs," in *DATE*, 2014.

[69] A. Das, B. M. Al-Hashimi, and G. V. Merrett, "Adaptive and hierarchical runtime manager for energy-aware thermal management of embedded systems," *TECS*, 2016.

[70] C. Bolchini, M. Carminati, A. Miele, A. Das, A. Kumar *et al.*, "Run-time mapping for reliable many-cores based on energy/performance trade-offs," in *DFTS*, 2013.

[71] P. J. Roussel *et al.*, "New methodology for modelling MOL TDDB coping with variability," in *IRPS*, 2018.

[72] R. Gao *et al.*, "NBTI-generated defects in nanoscaled devices: fast characterization methodology and modeling," *TED*, 2017.

[73] X. Wan *et al.*, "HCI improvement on 14nm FinFET io device by optimization of 3D junction profile," in *IRPS*, 2019.

[74] A. Balaji *et al.*, "PyCARL: A PyNN interface for hardware-software co-simulation of spiking neural network," in *IJCNN*, 2020.

[75] S. Song, A. Balaji, A. Das, N. Kandasamy *et al.*, "Compiling spiking neural networks to neuromorphic hardware," in *LCTES*, 2020.

[76] A. Balaji, S. Song, A. Das, J. Krichmar, N. Dutt *et al.*, "Enabling resource-aware mapping of spiking neural networks via spatial decomposition," *ESL*, 2020.

[77] A. Balaji, A. Das, Y. Wu, K. Huynh *et al.*, "Mapping spiking neural networks to neuromorphic hardware," *TVLSI*, 2020.

[78] A. Balaji and A. Das, "A framework for the analysis of throughput-constraints of SNNs on neuromorphic hardware," in *ISVLSI*, 2019.

[79] A. Das and A. Kumar, "Dataflow-based mapping of spiking neural networks on neuromorphic hardware," in *GLSVLSI*, 2018.

[80] A. Balaji, T. Marty, A. Das, and F. Catthoor, "Run-time mapping of spiking neural networks to neuromorphic hardware," *JSPS*, 2020.

[81] T. Titirsha, S. Song, A. Balaji, and A. Das, "On the role of system software in energy management of neuromorphic computing," in *Computing Frontiers*, 2021.

[82] A. Balaji and A. Das, "Compiling spiking neural networks to mitigate neuromorphic hardware constraints," in *IGSC Workshops*, 2020.

[83] A. Das, Y. Wu, K. Huynh *et al.*, "Mapping of local and global synapses on spiking neuromorphic hardware," in *DATE*, 2018.

[84] M. A. Khanesar, M. Teshnehlab, and M. A. Shoorehdeli, "A novel binary particle swarm optimization," in *Mediterranean Conference on Control & Automation*, 2007.

[85] A. Das, P. Pradhapan, W. Groenendaal, P. Adiraju, R. Rajan *et al.*, "Unsupervised heart-rate estimation in wearables with Liquid states and a probabilistic readout," *Neural Networks*, 2018.

[86] A. Das, F. Catthoor, and S. Schaafsma, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ECG," in *CHASE*, 2018.

[87] A. Balaji, F. Corradi *et al.*, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *JOLPE*, 2018.

[88] E. J. Moyer and A. Das, "Motif identification using cnn-based pairwise subsequence alignment score prediction," *arXiv preprint arXiv:2101.08385*, 2021.

[89] E. J. Moyer, A. Das *et al.*, "Machine learning applications to dna subsequence and restriction site analysis," *arXiv preprint arXiv:2011.03544*, 2020.

[90] T. Titirsha and A. Das, "Reliability-performance trade-offs in neuromorphic computing," in *IGSC Workshops*, 2020.

[91] T. Titirsha and A. Das, "Thermal-aware compilation of spiking neural networks to neuromorphic hardware," in *LCPC*, 2020.

[92] T. Titirsha, S. Song, A. Das, J. Krichmar, N. Dutt *et al.*, "Endurance-aware mapping of spiking neural networks to neuromorphic hardware," *TPDS*, 2021.

[93] D. L. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems: Beyond learning algorithms," in *AAAI*, 2013.

[94] G. W. Burr, M. J. Brightsky, A. Sebastian *et al.*, "Recent progress in phase-change memory technology," *JETCAS*, 2016.