Towards Functionally Robust AI Accelerators

Sanmitra Banerjee*, Ching-Yuan Chen*, Jonti Talukdar*, Shao-Chun Hung*, Arjun Chaudhuri*,
Mahdi Nikdast†, and Krishnendu Chakrabarty*
*Department of Electrical and Computer Engineering, Duke University
†Department of Electrical and Computer Engineering, Colorado State University

2021 IEEE Microelectronics Design & Test Symposium (MDTS) | 978-1-6654-2480-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/MDTS52103.2021.9476140

***Abstract*—Recent advances in deep learning can be attributed to the continued performance improvement of hardware processors and artificial intelligence (AI) accelerators. In addition to conventional CMOS accelerators based on Von Neumann architecture, emerging technologies such as silicon photonics, memristors, and monolithic 3D (M3D) integration are being explored as post-Moore's law alternatives. However, the energy efficiency and performance of emerging AI accelerators can be catastrophically impacted by faults due to fabrication-process variations, thermal crosstalk, and aging. In this paper, we analyze the performance of several emerging AI accelerators in the presence of different uncertainties, and present low-cost methods to assess the significance of faults and mitigate their effects. We show that across all technologies, the impact of uncertainties on the performance can vary significantly based on the fault type and the parameters of the affected component. Therefore, the fault criticality-assessment techniques presented in this paper are necessary for yield improvement.**

I. Introduction

The rapid growth in big data from mobile, Internet of things (IoT), and edge devices and the continued demand for higher computing power have established deep learning as the cornerstone of most artificial intelligence (AI) applications today. Recent years have seen a push towards deep learning implemented on domain-specific AI accelerators that support custom memory hierarchies, variable precision, and optimized matrix multiplication [1]. Commercial AI accelerators have shown superior energy and footprint efficiency compared to GPUs for a variety of inference tasks [2].

CMOS-based accelerators are, however, approaching fundamental bottlenecks because of (i) the slowdown in CMOS scaling leading to limited performance-per-watt, and (ii) low-bandwidth metallic interconnects. To address such issues, several emerging technologies are being explored. For example, optical interconnects represent an alternative to metallic interconnects with a promise of lower power consumption and latency and higher bandwidth [3]. Integrated photonic neural networks based on silicon photonics use optical interconnects and photonic components to perform matrix multiplication; doing so can reduce the computation time from $O(N^2)$ to $O(1)$ [4]. Deep neural networks (DNNs) are also being mapped to memristor devices due to their scalability, high performance, and non-volatility [5]. As for emerging integration solutions, monolithic 3D (M3D) stacking of different elements (e.g., processing and memory) is being explored [6] to compensate for large interconnect delays in AI accelerators.

Despite continued scaling in the nanometer technology nodes, process variations and manufacturing defects are inevitable, leading to permanent faults and soft errors. AI accelerators based on emerging technologies are prone to several reliability issues stemming from fabrication-process variations, thermal crosstalk, and aging-related uncertainties. In this paper, we discuss these roadblocks that need to be understood and analyzed to ensure functional robustness in emerging AI accelerators. In Sections II–V, we explore DNNs based on systolic arrays, memristors, silicon photonics, and M3D ICs. Moreover, we present various technology-specific low-cost fault-tolerance and criticality assessment techniques. Finally, we summarize the reliability concerns in emerging AI accelerators and draw conclusions in Section VI.

II. Von Neumann CMOS Accelerators

A. Need for Robustness Analysis

Prior work [7] suggests that DNNs are inherently tolerant to noisy inputs due to the robust training process. Moreover, regularization features such as dropout, normalization, and loss minimization [8] make DNNs robust to permanent faults and soft errors [9]. Robustness analysis identifies faults (i.e., stuck-at faults, delay faults, and transient faults) that result in significant deviations from system specifications.

The robustness of a systolic array-based accelerator in presence of faults depends on the fault locations, the dataflow through the array, the mapping of the DNN model to the array, and the parameter distribution of the DNN. Recent work on systolic array accelerator architectures carried out the analysis of functional errors caused by faults injected in the I/O pins of a processing element (PE) in the systolic array [10], [11]. Deep learning (DL) driven methods for criticality assessment of stuck-at faults in gate-level PE netlists have been presented in [12] and [13].

B. Analysis Using Deep Learning

Training ML models for criticality classification requires feature extraction based on the topology of the circuit, the functional dataflow through the design, and functional features like gate type. In [12], we utilize DNN-based classifiers and the features described above for assessing the functional criticality of stuck-at faults in the synthesized netlists of 128×128 systolic arrays implemented using 16-bit and 32-bit floating-point architectures. We map the LeNet-5 neural network to the systolic array-based accelerator for carrying out inferencing on a dataset containing 100 representative MNIST images. The functional criticality of a fault is determined by its impact on the fault-free inferencing accuracy. If both stuck-at-0 (s-a-0) and stuck-at-1 (s-a-1) faults at a gate's output are functionally

978-1-6654-2480-6/21/$31.00 ©2021 IEEE

TABLE I: Evaluation of aggregated GAN-based fault-criticality assessment in 32-bit PE(20,0) and 16-bit PE(20,0).

Exp.	Adder in 32-bit PE							Multiplier in 32-bit PE							16-bit PE						
	N_{cr}	A_{ML1} (%)	F_{cr}	TE_1	TE_2	TE_2^p (%)	$\boldsymbol{TE_{cat}}$ **(%)**	N_{cr}	A_{ML1} (%)	F_{cr}	TE_1	TE_2	TE_2^p (%)	$\boldsymbol{TE_{cat}}$ **(%)**	N_{cr}	A_{ML1} (%)	F_{cr}	TE_1	TE_2	TE_2^p (%)	$\boldsymbol{TE_{cat}}$ **(%)**
I	109	89.4	139	7	7	5.0	**2.8**	52	90.0	55	12	4	7.2	**3.6**	111	60.4	158	45	3	1.8	**0**
II	114	89.2	142	8	8	5.6	**2.1**	48	90.7	51	11	4	7.8	**3.9**	105	61.4	149	49	3	2.0	**0**
III	114	88.9	143	7	7	4.8	**2.0**	45	90.1	59	9	2	3.3	**4.3**	110	61.4	153	51	3	1.9	**0**

N_{cr}: number of critical nodes in the evaluation set; A_{ML1}: accuracy of ML1 on evaluation set; F_{cr}: number of critical faults in the evaluation set; TE_2^p: $(TE_2/F_{cr}) \times 100$; TE_{cat}: percentage of faults misclassified as benign by ML2, causing more than 20% drop in inferencing accuracy.

benign, the gate is labeled as functionally benign; otherwise, it is labeled as critical.

We develop a two-tier DNN-based model to classify functional fault criticality. The first tier model (ML1) classifies a node as either critical or benign. However, the challenge with fault-criticality assessment using a single tier (i.e., ML1) is the inevitability of misclassification; even a highly effective model can lead to misclassification. We use a second ML model (ML2) to identify test-escapes from the predictions by ML1.

Training and validation: Ground-truth data comprising the criticality information of nodes in a gate-level PE is collected for training and validation of ML1. Single fault simulation is carried out to determine the functional criticality of a node based on a pre-determined threshold. We set a conservative threshold of 95% inferencing accuracy for collecting more critical-labeled nodes. First, the ML1 is trained to classify nodes as critical or benign. The trained model is then evaluated on the validation set. The critical nodes in the validation set that are misclassified as benign by ML1 are added to the set of misclassified nodes S_{TE}. The set S_{TE} is used for training a generative adversarial network (GAN [14]) to generate samples with features matching the feature distribution of the actual misclassified nodes. This ML scheme is referred to as aggregated GAN-based criticality assessment. The trained GAN is then used for generating test escape-like samples to further augment S_{TE}. The augmented S_{TE}, along with actually benign nodes, is used to train ML2.

Evaluation: The pre-trained ML1 model is used to predict the criticality of nodes not seen before during training. The nodes predicted as benign by ML1 are fed to the pre-trained ML2 model for post-processing—ML2 evaluates if any of those nodes is misclassified. Finally, the nodes classified as real benign by ML2 are categorized as the truly-benign nodes. Table I shows the evaluation results for the 32-bit adder and multiplier, and 16-bit PE. Note that TE_1 (TE_2) denotes the number of test-escape faults after prediction by ML1 (ML2).

C. Analysis Using Graph Convolutional Networks (GCNs)

A GCN leverages the topology of a graph for the classification of nodes in the graph [15]. The gate-level netlist of the PE can be represented as a directed graph G, where the nodes represent gates and edges represent interconnections. A GCN-based 2-tier criticality analysis framework is presented in [13].

Training and validation: The first tier applies a GCN model, referred to as GCN-1, to classify the criticality of a node. The labeled set of nodes S_{GT} is randomly split into training and validation sets; r_{tr} is the fraction of nodes in S_{GT} assigned

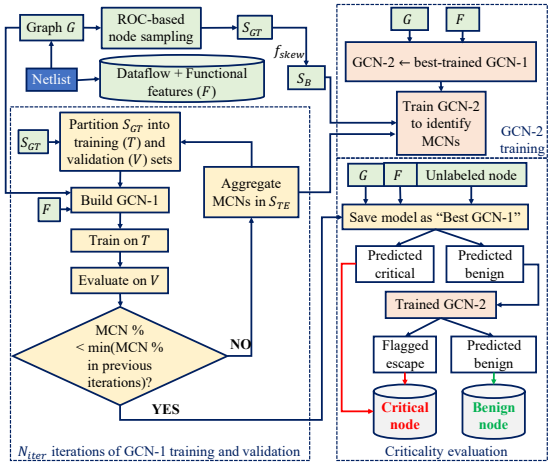


Fig. 1: Training and evaluation of a 2-tier GCN-based framework.

to the training set. The adjacency matrix of G, functional and data flow-based features of all nodes in G, and the criticality labels of the nodes in the training set are used to train GCN-1. The GCN-1 model uses a multi-layer perceptron for criticality classification. The trained GCN-1 is validated on the nodes in the validation set. During validation, the GCN-1 may misclassify some critical nodes as benign. At the same time, some benign nodes may be misclassified as critical; such a scenario is considered to be a *false alarm*. We follow a conservative approach by prioritizing the minimization of critical-fault misclassification (CFM). To reduce CFMs, the second tier uses another GCN model, referred to as GCN-2, to learn the feature distribution of the critical nodes misclassified by GCN-1 and distinguish them from the benign nodes. The misclassified critical nodes (MCNs) obtained during the validation of GCN-1 are added to a set, S_{TE}. The nodes in S_{TE}, along with benign nodes classified correctly by GCN-1, are used to train GCN-2. Fig. 1 illustrates the 2-tier GCN-based framework.

Evaluation: During the evaluation of the functional criticality of the unlabeled nodes in G, the nodes classified as benign by GCN-1 are evaluated by GCN-2 for the potential detection of test-escapes. If a node is classified as critical by either GCN-1 or GCN-2, it is considered to be functionally critical. Table II presents the evaluation results of the framework for the 32-bit adder and multiplier of PE(20,0), and for 16-bit PE(20,0). The percentage reduction in the number of faults to be targeted for in-field testing is denoted by Δ_S. Here, $\Delta_S = 100 \cdot N_B/N_T$, where N_B is the number of faults classified as benign and N_T is the total number of faults in the netlist.

TABLE II: Performance summary of 2-tier GCN-based framework.

Netlist	A_c (%)	M_c (%)	N_T	N_B	Δ_S (%)
Add32	81.2	1.2	6952	4562	65.6
Mult32	84.4	0	6525	5737	87.9
PE16	78.8	0	6415	2477	38.6

III. Memristor-Mapped Deep Neural Network

A promising implementation pathway for DNN models is to map them to specialized neuromorphic hardware, e.g., memristor-based crossbars. However, recent research has highlighted a number of reliability concerns when DNN models are mapped to memristor-based crossbars. Crossbar faults can deviate the mapped DNN weights from their nominal values.

A. Memristor Crossbars and Weight Mapping

Memristors can be viewed as programmable resistors. A memristor crossbar can perform vector-matrix multiplication with *O(1)* computation time and without the need for data movement between processors and memory. Fig. 2(a) shows a crossbar of memristors that can be used for vector-matrix multiplication. A memristor crossbar that is implemented using resistive random-access memory (ReRAM) cells can achieve extremely high density, as ReRAM cells are 10-times smaller than DRAM cells [16]. Micron and Sony have fabricated a 16 Gb 8192×2048 ReRAM crossbar [17]; each cell corresponds to a single bit and can store a high or low conductance level.

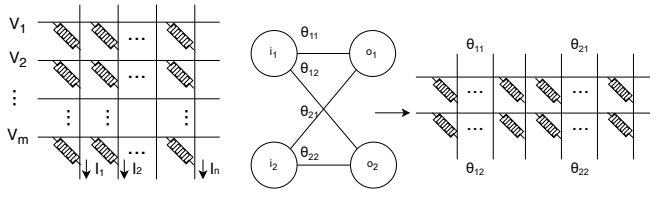
In general, a realistic DNN model has tens of millions of weights: e.g., AlexNet has 61 million weights and uses a total of 6.1 Gb memristor cells. Such a large number of memristors implemented using ReRAM cells can achieve extremely-high memory cell density (10 times the cell density of DRAM [16]) and does not require excessive on-chip area.

B. Crossbar Fault Models

Faults in a crossbar result in the deviation of nominal DNN-model weights. These faults lead to deviations (referred to as matrix $\mathbf{\Delta}$ in this paper) of the DNN-model weights (referred to as matrix $\mathbf{\Theta}$ in this paper) from the nominal values; the DNN weights in the faulty crossbar thus become $\mathbf{\Theta} + \mathbf{\Delta}$. We next review some common fault models for a memristor crossbar.

Stuck-on/off faults: A memristor cell can be stuck-on or stuck-off, i.e., its conductance is stuck at either g_{on} (stuck-on) or g_{off} (stuck-off). In our quantization setting, a stuck-on (off) cell forces the corresponding weight bit to be 1 (0). Several other failure mechanisms in memristors can be modeled as stuck-on/off faults [19]. One such example is a deep fault where the conductance of a memristor is affected after a large number of reading (inference) operations.

Stuck-open/short faults [20]: A stuck-open/short fault forces the conductance of a memristor cell to fall out of its nominal range. A stuck-open fault implies that the memristor cell is open, i.e., $g_{\text{open}} \in [0, g_{\text{off}}]$. On the other hand, a short fault, i.e., $g_{\text{short}} \in [g_{\text{off}}, \infty)$, induces a large current through the corresponding crossbar column and it can damage the circuit. Therefore, an open/short fault can force the associated weight to go beyond the nominal range $[\theta_{\min}, \theta_{\max}]$.



(a) (b)

Fig. 2: (a) Illustration of a memristor crossbar; (b) An example to illustrate the mapping of a fully-connected layer [18].

Slow-write faults: A defective memristor may require a longer write cycle to overwrite the conductance value. This is referred to as a slow-write fault [21]. Similar to a deep fault, a slow-write fault can be the direct consequence of consecutive write—1/0 operations. In addition, deviations in physical parameters can cause slow-write faults.

Read/write disturbance and coupling faults: While reading (writing into) a memristor, the read (write) current may affect other memristors in the same column [22], [23]. Similar to the write disturbance, writing a memristor can affect the nearby "victim" memristors. A crossbar with a high density of memristor cells is prone to coupling faults [19].

C. Fault-Tolerance in Memristor Crossbars

To ensure fault tolerance, techniques that detect, locate, and recover from faults have been proposed for memristor-based crossbars. Fault detection [24], [25] and localization [26] can be followed by mitigation methods such as remapping and retraining to restore the classification accuracy [27]–[29]. March-based test algorithms can accurately detect and locate deviated weights, but they impose high test time [25]. [30] used adversarial examples for functional testing to detect weight deviations in memristor-mapped DNNs to reduce the test time. However, the approach in [30] still relies on time-consuming March test algorithms to localize deviations and restore the deviated weights to the desired values.

IV. Silicon-Photonic Neural Networks (SPNNs)

Silicon-photonic neural networks (SPNNs) use photonic components to realize linear operations (i.e., matrix multiplication) with ultra-high speed and ultra-low energy consumption [31]. The linear multipliers can be represented using two unitary multipliers and a diagonal matrix, which are obtained using singular value decomposition (SVD). The multipliers and the diagonal matrix can be realized using a network of interconnected Mach–Zehnder interferometers (MZIs) [32], a common silicon photonic device, each of which includes beam splitters (BeS) and phase shifters (PhS) to split and change the phase angle of optical signals (see Fig. 3). Deviations in the phase angles in PhS (ϕ and θ in Fig. 3) and the splitting ratio in BeS–due to inevitable fabrication-process variations and thermal crosstalk–have a critical impact on SPNN performance. We systematically analyze the impact of such uncertainties in SPNNs in a hierarchical fashion [33].

A. Hierarchical Study of Uncertainties in SPNNs

Component-Level (Phase Shifters and Beam Splitters): The phase change in thermo-optic PhS is proportional to the temperature change applied using thermal actuators (i.e., microheaters). The proportionality constant can be affected by lithographic variations. Additionally, mutual thermal crosstalk

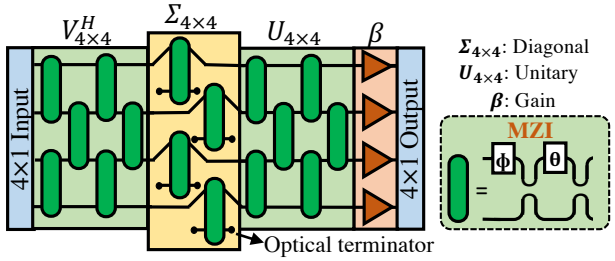


Fig. 3: Linear-layer representation using MZI arrays. A 4×4 linear layer is shown here.

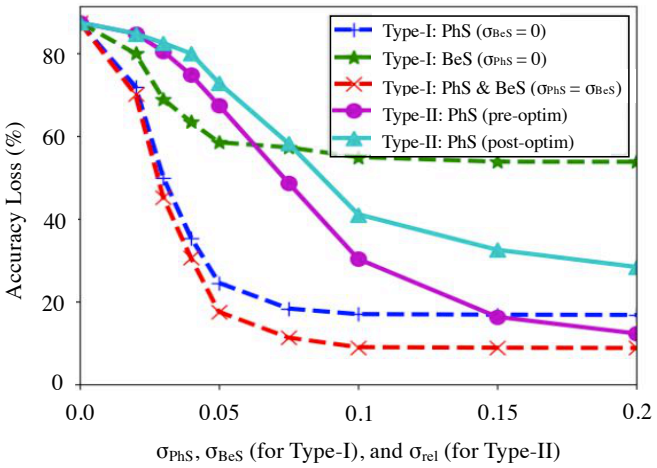


Fig. 4: Accuracy loss due to random type-I (nominal independent, dotted lines) uncertainties in PhS and BeS, and type-II (nominal-dependent, solid lines) uncertainties in PhS.

among neighboring devices affects the efficiency of the tuning and bias-control mechanism, imposing phase-angle errors. Prior studies indicate an average fabrication error of ≈0.21 radian in the tuned phase angles in PhS [34]. Uncertainties in phase angles can be of two types: type-I: nominal-dependent uncertainties, where the variation in a phase shifter is expected to be proportional to the nominal tuned phase angle (e.g., thermal crosstalk), and type-II: nominal-independent uncertainties, where the variation is independent of the nominal phase angle (e.g., process variations). For fabricated BeS, prior work shows a 1–2% deviation from 50:50 ideal splitting ratio [34]. Note that uncertainties in BeS are exclusively type-I as the nominal splitting ratio is 50:50 for all BeS.

Device-level (MZIs): The 2×2 MZI transfer matrix, T_{MZI}, is a function of the tuned phase angles θ and ϕ. Variations in these phase angles in PhS result in deviations in T_{MZI}, leading to faulty MZI operation. For nominal-dependent uncertainties, this deviation in the T_{MZI} is catastrophic for MZIs with high PhS angles [33] [35]. Similarly, T_{MZI} also varies due to uncertainties in the BeS splitting ratios.

Layer-level (MZI Array): The unitary multipliers in the linear layers of an SPNN can deviate from their intended form due to MZI uncertainties in an array. In [33], we insert such uncertainties in different MZIs and show that the relative-variation distance between the intended and deviated unitary multipliers depends on the location and the nominal phase angles of the affected MZI. Variations in each MZI have a unique impact as different elements of the unitary multiplier are affected by different subsets of MZIs in the array.

System-level (SPNN): Variations in the MZI arrays lead to faulty matrix multiplication, thereby imposing accuracy losses in the SPNNs. To demonstrate this, we train a fully-connected SPNN with two hidden layers of 16 complex-valued neurons using the MNIST dataset. During inferencing, we insert nominal-independent uncertainties in (i) only PhS, (ii) only BeS, and (iii) both PhS and BeS. The uncertainties are sampled from a zero-mean Gaussian distribution with standard deviations of $\sigma_{PhS} \cdot 2\pi$ (for PhS) and $\sigma_{BeS} \cdot 1/\sqrt{2}$ (for BeS). The red dashed line in Fig. 4 shows that under type-I uncertainties, the mean classification accuracy over 1000 Monte Carlo iterations decreases steeply with σ till the accuracy drops below 10% (random guess) at $\sigma \approx 0.075$. The solid magenta line in Fig. 4 shows the impact of type-II uncertainties in PhS. In this case, the standard deviation of the uncertainties is given by $\sigma_{rel} \cdot \mu$, where μ denotes the nominal phase angle in PhS.

B. Optimizing SPNNs under Random Uncertainties

The tuning-power consumption and the susceptibility to nominal-dependent uncertainties increase with the phase angles in thermo-optic PhS. We leverage the non-uniqueness of SVD under reflections to realize the linear multipliers with minimal phase angles, thus improving the power efficiency and robustness in SPNNs [35]. When applied to a fully connected SPNN, our method leads to an average reduction of 12.8% in the overall network phase angles and consequently, the tuning power consumption (with up to 15.3% reduction in the phase angles in one layer). The solid cyan line in Fig. 4 shows that the optimized SPNN leads to a lower accuracy loss and improvement in robustness under type-II uncertainties in the PhS. The improvement in robustness increases at higher levels of uncertainties (e.g., at $\sigma_{rel} = 0.2$, the accuracy loss is reduced by 16.1%). Note that under this optimization, the trained weight matrix remains unchanged.

V. Monolithic 3D Integration

Monolithic 3D (M3D) integration leverages fine-grained monolithic inter-tier vias (MIVs) for the connections between tiers. Because MIVs are similar to conventional back-end-of-line vias, they can achieve high reliability and high integration density, providing significant improvements in wirelength reduction and interconnect power consumption. Therefore, M3D integration has been proposed as a promising solution for AI accelerators.

A. M3D-integrated Memory Systems for AI Accelerators

[41] proposes an M3D nonvolatile RAM (NVRAM) interface to alleviate memory-bounded problems during training. In this architecture, the bottom-most tier is devoted to components for in-memory computation. The second tier is the memory controller tier, on top of which are resistive random access memory (RRAM) tiers. M3D integration enables memory buses to be 1 KB wide because of the advantages of MIVs. Such memory buses significantly reduce the access latency and improve energy efficiency. Furthermore, the read and write

TABLE III: Summary of prior work on analysis and mitigation of reliability issues in AI accelerators.

Reference	Architecture	Technology	Fault Model	Results	
				Analysis	Mitigation
[12]	LeNet-5	CMOS-based Von Neumann systolic-array	stuck-at	90% accuracy loss due to critical faults and up to 5% accuracy loss due to benign faults.	Fast fault-criticality assessment using 2-tier GAN with negligible test escape.
[13]	LeNet-5	CMOS-based Von Neumann systolic-array	stuck-at	90% accuracy loss due to critical faults and up to 5% accuracy loss due to benign faults.	Fault-criticality assessment with low feature-extraction effort compared to [12].
[10]	DNNs and AlexNet	CMOS-based Von Neumann systolic-array	stuck-at	35% TIMIT accuracy loss with 0.005% faulty MACs	Up to 50% faulty MACs can be tolerated with fault-aware pruning and retraining.
[36]	DNNs and AlexNet	Von Neumann systolic-array	stuck-at, transient	Up to 90% accuracy loss due to 0.0003% fault rate.	Test patterns for functional safety assessment achieves an average of 92.63% fault coverage.
[7]	VGG, LeNet, TiGRU, ResNet-50	Algorithmic DNN	static memory faults	Activations more robust to faults than weights, weight/activation reuse determines robustness.	–
[9]	AlexNet, CaffeNet, NiN, ConvNet	Von Neumann systolic-array	transient faults	Higher-order bits vulnerable to silent-data corruptions, normalization layers in DNN mitigate fault impact.	–
[30]	Multi-layer peerceptron, LeNet, ConvNet	Memristors	stuck-at faults in the crossbar	8% accuracy loss due to permanent and soft faults.	Edge-cloud computing framework detects up to 98% crossbar faults.
[28]	2-layer DNN	Memristors	stuck-at faults in the crossbar	MNIST accuracy drops to 42.5% under 20% single-bit failure (SBF) rate.	98.1% accuracy under 20% SBF with selective retraining of critical weights using defect information.
[37]	Fully-connected DNN (Reck architecture [38])	SPNN (MZIs)	phase encoding error and photodetection noise	76.7% accuracy (compared to 91.7% with software).	–
[33]	Fully-connected DNN (Clement architecture [32])	SPNN (MZIs)	random and localized uncertainties in PhS and BeS	70% loss in accuracy, impact of uncertainties depend on nominal parameter value and fault location, PhS more critical than BeS	–
[39]	2-layer Fully-connected DNN, LeNet-5, and augmented LeNet-5	SPNN (MZIs)	random uncertainties in PhS	Fully-connected: 11%, LeNet-5: 11%, and Aug.LeNet-5: 13%	Regularization term to minimize the phase angles added to the training cost function. Fully-connected: 97%, LeNet-5: 75%, Aug. LeNet-5: 82%
[40]	Fully-connected DNN (GridNet and FFTNet architecture)	SPNN (MZIs)	random uncertainties in PhS and BeS	MNIST accuracy of GridNet drops to 50% under uncertainties (compared to 98% in fault-free case)	FFTNet maintains near-constant performance under uncertainties.
[35]	Fully-connected DNN (Clement architecture [32])	SPNN (MZIs)	random and localized uncertainties in PhS and BeS	Up to 70% accuracy loss under nominal-dependent (type-II) uncertainties in PhS	SVD-based zero-cost optimization reduces accuracy loss by 16.1% and tuning power consumption by 15.3%.

accesses can be separated by adding another set of MIVs. This can prevent the write access from blocking the read access and therefore increase the memory bandwidth. Experimental results show that the accelerator in [41] outperforms a state-of-the-art GPU in performance, power consumption, and energy efficiency during both training and inference.

In [6], an RRAM-based CNN accelerator is proposed. With heterogeneous M3D architecture—multiple technology nodes for different tiers—the accelerator can achieve scaling of peripheral logic while RRAM cells remain at legacy nodes. [6] places RRAM arrays and CMOS transistors, which require high programming voltages, in the top tier with 40 nm process; the bottom tier is composed of digital logic and analog-to-digital converters (ADCs) with the advanced 28/16 nm nodes. The scaling of the bottom tier spares extra spaces for additional ADCs, which improves the throughput and energy efficiency of RRAM accelerators.

B. Robustness During M3D Testing

The benefits of M3D integration are accompanied by new challenges. Power-supply noise (PSN) is one of the major concerns due to high current demand in the upper tiers. The voltage droop induced by PSN may cause erroneous results during circuit operation. The testing mode suffers more from PSN than the functional mode because of excessive switching activity. The additional delay from PSN-induced voltage droop may lead to fault-free circuits failing on the tester (yield loss).

In [42], an ILP-based reshaping algorithm for M3D transition-delay fault patterns is presented. The authors create a new analysis framework specific for M3D designs to obtain PSN-induced voltage droop of each cell. A detailed analysis between switching activities and voltage droop shows that switching activities in the top tier are highly related to the overall voltage droop. An ILP-based algorithm is proposed to reshape such patterns to minimize switching activities in the top tier. Patterns after reshaping have been shown to not only eliminate the yield-loss problem but also reduce the slack margin.

C. Challenges in M3D Integration

Prior work on M3D-integrated AI accelerators is based on the assumption that there is no process variation among tiers. However, one of the major challenges of M3D is to

fabricate upper tiers without damaging components underneath. Interconnect and transistors in the bottom tiers cannot withstand the standard thermal budget for ion implantation and annealing when implementing upper-tier transistors. Therefore, low-temperature processing technologies need to be developed. Solid-phase epitaxy regrowth (SPER) and laser annealing have been demonstrated to successfully realize top-tier transistors without damaging lower-tier transistors. Unfortunately, SPER may have high source-drain resistance while laser annealing results in low on-current. Both techniques cause performance degradation for transistors in the top tiers. Such degradation negates the benefits of M3D integration. This issue needs to be addressed for M3D designs before commercial exploitation.

Furthermore, metal usage of intermediate back-end-of-line (iBEOL) between each tier is another concern. The processing temperature of SPER and laser annealing is not low enough to utilize traditional copper interconnect in the bottom tiers without damages. Tungsten is an alternative for iBEOL, but its bulk resistance is $3.1\times$ higher than the resistance of copper. The increase in resistance results in extra latency of bottom tiers. This diminishes the advantages of M3D-integrated AI accelerators because bottom tiers are mainly dedicated to in-memory computing components. Therefore, although M3D integration is a promising solution for AI accelerators, research effort is needed to mitigate performance degradation issues.

VI. Discussion and Conclusion

While state-of-the-art AI accelerators have shown great promise in implementing computationally expensive deep-learning algorithms, their susceptibility to design-time and run-time uncertainties remains a concern. Table III summarizes prior work on the reliability issues in emerging AI accelerators discussed in this paper. The underlying theme, across all prior work, is the necessity of robustness assessment and allied low-cost fault mitigation techniques. The individual challenges of the different fabrication techniques will be compounded by those for M3D stacking, thereby making fault-criticality assessment and optimization even more crucial for heterogeneously integrated systems.

References

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

[2] Y. E. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking TPU, GPU, and CPU platforms for deep learning," *arXiv preprint arXiv:1907.10701*, 2019.

[3] F. P. Sunny *et al.*, "A survey on silicon photonics for deep learning," *arXiv preprint arXiv:2101.01751*, 2021.

[4] R. A. Athale, , and W. C.Collins, "Optical matrix–matrix multiplier based on outer product decomposition," *Applied optics*, vol. 21, no. 12, pp. 2089–2090, 1982.

[5] H. Jeong and L. Shi, "Memristor devices for neural networks," *Journal of Physics D: Applied Physics*, vol. 52, no. 2, p. 023003, 2018.

[6] G. Murali, X. Sun, S. Yu, and S. K. Lim, "Heterogeneous mixed-signal monolithic 3-D in-memory computing using resistive RAM," *TVLSI*, pp. 386–396, 2021.

[7] B. Reagen et al., "Ares: A framework for quantifying the resilience of deep neural networks," in *DAC*, 2018.

[8] P. Baldi et al., "Understanding dropout," in *NeurIPS*, 2013.

[9] G. Li et al., "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *ACM SC*, 2017.

[10] J. J. Zhang et al., "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *VLSI Test Symposium*, 2018.

[11] Z. Xu et al., "Safety Design of a Convolutional Neural Network Accelerator with Error Localization and Correction," in *ITC*, 2019.

[12] A. Chaudhuri et al., "Functional criticality classification of structural faults in AI accelerators," in *ITC*, 2020.

[13] A. Chaudhuri et al., "Fault-criticality assessment for AI accelerators using graph convolutional networks," in *DATE*, 2021.

[14] A. Radford et al., "Unsupervised representation learning with deep convolutional generative adversarial networks," *ICLR*, pp. 1–16, 2015.

[15] T. N. Kipf et al., "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[16] H.-S. P. Wong et al., "Metal–Oxide RRAM," *Proc. IEEE*, vol. 100, pp. 1951–1970, 2012.

[17] R. Fackenthal et al., "A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology," in *ISSCC*, 2014.

[18] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE JETCAS*, vol. 9, pp. 570–579, 2019.

[19] A. Chaudhuri and K. Chakrabarty, "Analysis of process variations, defects, and design-induced coupling in memristors," in *ITC*, 2018.

[20] S. Hamdioui et al., "Testing open defects in memristor-based memories," *IEEE Trans. Computers*, vol. 64, pp. 247–259, 2015.

[21] S. Kannan et al., "Sneak path testing and fault modeling for multilevel memristor-based memories," in *ICCD*, 2013.

[22] C. Chen et al., "RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE Trans. Computers*, vol. 64, pp. 180–190, 2015.

[23] Y. Chen and J. Li, "Fault modeling and testing of 1T1R memristor memories," in *IEEE VLSI Test Symposium (VTS)*, 2015.

[24] M. Fieback et al., "Device-aware test: A new test approach towards DPPB level," in *ITC*, 2019.

[25] S. Kannan et al., "Sneak-path testing of crossbar-based nonvolatile random access memories," *IEEE Trans. Nanotechnology*, vol. 12, pp. 413–426, 2013.

[26] Z. Xu and J. Abraham, "Safety design of a convolutional neural network accelerator with error localization and correction," in *ITC*, 2019.

[27] B. Li et al., "ICE: Inline calibration for memristor crossbar-based computing engine," in *DATE*, 2014.

[28] C. Liu et al., "Rescuing memristor-based neuromorphic design with high defects," in *DAC*, 2017.

[29] L. Xia et al., "Stuck-at fault tolerance in RRAM computing systems," *IEEE J. Emerg. Sel. Topics in Circuits and Systems*, 2018.

[30] W. Li et al., "RRAMedy: Protecting ReRAM-based neural network from permanent and soft faults during its lifetime," in *ICCD*, 2019.

[31] Q. Cheng *et al.*, "Silicon photonics codesign for deep learning," *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1261–1282, 2020.

[32] W. R. Clements *et al.*, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, no. 12, pp. 1460–1465, 2016.

[33] S. Banerjee *et al.*, "Modeling silicon-photonic neural networks under uncertainties," *IEEE/ACM DATE*, 2021.

[34] F. Flamini *et al.*, "Benchmarking integrated linear-optical architectures for quantum information processing," *Scientific Reports*, 2017.

[35] S. Banerjee *et al.*, "Optimizing coherent integrated photonic neural networks under random uncertainties," to appear in *IEEE/OSA OFC*, 2021.

[36] S. Kundu et al., "Toward functional safety of systolic array-based deep learning hardware accelerators," *TVLSI*, 2021.

[37] Y. Shen *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature Photonics*, vol. 11, no. 7, p. 441, 2017.

[38] M. Reck *et al.*, "Experimental realization of any discrete unitary operator," *Physical Review Letters*, vol. 73, no. 1, p. 58, 1994.

[39] Y. Zhu, G. L. Zhang, B. Li, X. Yin, C. Zhuo, H. Gu, T.-Y. Ho, and U. Schlichtmann, "Countering variations and thermal effects for accurate optical neural networks," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–7.

[40] M. Y. S. Fang *et al.*, "Design of optical neural networks with component imprecisions," *Optics Express*, pp. 14 009–14 029, 2019.

[41] Y. Yu and N. K. Jha, "SPRING: A sparsity-aware reduced-precision monolithic 3D CNN accelerator architecture for training and inference," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2020.

[42] S. C. Hung, Y. C. Lu, S. K. Lim, and K. Chakrabarty, "Power supply noise-aware scan test pattern reshaping for at-speed delay fault testing of monolithic 3D ICs *," in *ATS*, 2020.