RoTA: Rotational Torus Accelerator for Wear Leveling of Neural Processing Elements

Taesoo Lim, Hyeonjin Kim, Jingu Park, Bogil Kim and William J. Song
School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea
{taesu.lim, hyeonjin_kim, jin9park, bogilkim, wjhsong}@yonsei.ac.kr

***Abstract*—This paper introduces a reliability-aware neural accelerator design with a wear-leveling solution that balances the utilization of processing elements (PEs). Neural accelerators deploy many PEs to exploit data-level parallelism, but their designs and operations have focused mostly on performance and energy efficiency metrics. Directional dataflows in PE arrays and dimensional misalignment with variable-sized neural layers cause the underutilization of PEs, which is biased to PE locations and gradually accumulated over time. Consequently, the accelerators experience severe usage imbalance between PEs. To resolve the problem, this paper proposes a *rotational torus accelerator* (RoTA) with an optimized wear-leveling scheme that shuffles PE utilization spaces to eliminate PE usage imbalance. Evaluation results show that RoTA improves lifetime reliability by 1.69x.**

I. Introduction

Deep neural network (DNN) accelerators employ a large number of processing elements (PEs) to exploit massive data-level parallelism. Accelerator designs and operations have been widely explored in performance and energy efficiency aspects [2], [15], [19], [20], [23], [25], but their reliability consequences were rarely studied. As neural accelerators are increasingly integrated into many reliability-critical systems (e.g., automotive, aerospace), their designs and operations should also be considered from a reliability standpoint.

A DNN accelerator processes a sizable neural layer by dividing it into tiles fitting into on-chip buffers and spatio-temporally reusing data across PEs. The PEs are typically arranged in a two-dimensional array, where computation starts from one corner of the PE array and propagates to other sides. However, layer dimensions do not align perfectly with the PE array size in most cases, even with deliberately optimized scheduling. Dimensional mismatches result in the underutilization of PEs, where some PEs do not engage in data processing. Initiating computations always from one corner of the PE array makes PEs near the starting point highly activated, while those at the opposite corner are rarely utilized. The usage imbalance is biased based on PE locations and gradually accumulates over time, threatening lifetime reliability.

To resolve the issue, this paper proposes a *rotational torus accelerator (RoTA)* with an optimized wear-leveling scheme that shuffles the starting point of data tile computations circularly in a PE array to balance the PE usage. We refer to a region of the PE array that engages in data processing as a *utilization space*. To avoid activating the same utilization space for many data tiles in a neural layer, the proposed wear-leveling scheme shifts the utilization space of data tiles using torus connections. It finds the least common multiple (LCM) of data tile and PE array widths, and utilization spaces are shifted horizontally for (LCM/tile_width) times. The PE array then becomes perfectly wear-leveled horizontally. The next utilization space moves vertically and repeats the same process of horizontal shifting. Hundreds of data tiles activate horizontally and vertically shifting utilization spaces in the torus-connected PE array, called *rotational wear-leveling (RWL)*.

The RWL method effectively eliminates usage imbalance between PEs except for a few activation spaces at the end of a layer due to a dimensional misalignment. If the next layer resets its utilization space to start over from the initial corner of the PE array, the PE usage difference at the end of every layer gradually accumulates. Thus, RWL is combined with *residual optimization (RO)* to continue shifting utilization spaces from the endpoint of the previous layer. It makes some PEs unused or reused when transitioning from one layer to another due to their size difference, but the torus network helps distribute such imbalance randomly across the entire PE array. In the long run, the PE usage imbalance is *unbiased* to PE locations and *non-cumulative* over time. Evaluation results show that RoTA with the RWL+RO scheme improves lifetime reliability by 1.69x on average for various DNN workloads [6], [7], [9], [18], [22], [24], [4], [12], [26] with only 0.3% design overhead and no performance degradation, compared to typical accelerator designs without wear-leveling [2], [19], [20], [23], [25].

II. Background

A DNN accelerator deploys a large number of processing elements for massive data-level parallelism, as shown in Fig. 1. PEs are typically organized in a 2-D array, where each PE contains a multiply-accumulate (MAC) unit and local buffers (LBs) to store input, weight, and output data [2], [19], [20], [23], [25]. To process the sizable data of a neural layer, a tiled dataset is fetched from off-chip memory to the on-chip global buffer (GLB) and distributed to PEs. The accelerator has two types of on-chip networks for data transfers: global and local networks. The global network connects the shared GLB to PEs for data scattering and gathering. The local network refers to unidirectional inter-PE connections for transferring partial sums [2], [20] or sharing input and weight values for spatial reuse [20]. The optimal execution flow and utilization space size of

This work was supported by the National Research Foundation of Korea (#RS-2023-00283799), the Korea Evaluation Institute of Industrial Technology (#RS-2023-00236772), and the Ministry of Science and ICT of Korea through the ITRC support program supervised by the Institute for Information and Communications Technology Planning and Evaluation (#IITP-2020-0-01847). William J. Song is the corresponding author.

978-3-9826741-0-0/DATE25/© 2025 EDAA

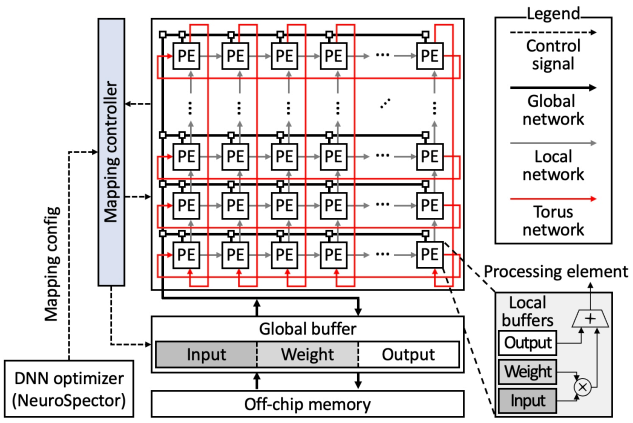
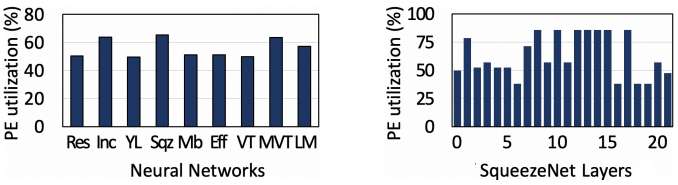


Fig. 1. RoTA adds torus connections to the PE array for wear-leveling in addition to existing global and local networks of a typical accelerator design.



(a) PE utilization of DNN workloads (b) PE utilization of Sqz layers

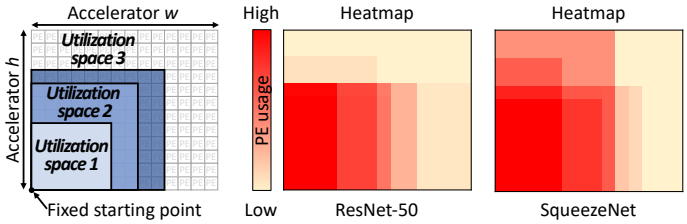
Fig. 2. (a) The energy-optimal execution of Eyeriss [2] utilizes only 55.8% of PEs on average. (b) Differently shaped neural layers [9] have drastically different PE utilization ratios.

the PE array vary depending on neural layer shapes, and they can be derived from scheduling optimizers for most energy-efficient or least-cycle executions [8], [14], [15].

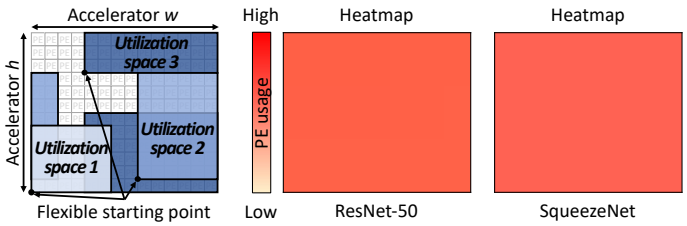
III. Motivation

DNN accelerators aim to maximize energy efficiency and performance by increasing data reuse in on-chip buffers and PE utilization (i.e., # of activated PEs). However, layer dimensions do not align perfectly with the PE array size, even with layer-wise optimized scheduling. Fig. 2a plots the average PE utilization of various DNN workloads on Eyeriss [2] for energy-optimal execution [15], revealing that the accelerator utilizes only 55.8% of PEs. Fig. 2b also shows that the PE utilization varies drastically within a neural network.

The existing accelerators initiate computation from a fixed starting point, such as the lower-left corner of the PE array in Fig. 3a. The fixed starting point and the underutilized PE array cause severe PE usage imbalance, as manifested in the utilization heatmaps of ResNet [6] and SqueezeNet [9] layers on the 14×12 PE array of Eyeriss [2]. As a result, PEs undergo substantially different stresses, and those experiencing greater stresses constrain the lifetime reliability of the accelerator. In Fig. 3a, the leftmost plot shows the utilization space of three selected layers from ResNet on Eyeriss. Utilization space 1 activates only 18% of PEs, and utilization spaces 2 and 3 use 38% and 54% of the PE array. Since utilization space 1 covers only a small part of the PE array, it can move to other regions to activate different PEs for wear-leveling but only to a limited extent due to a dimensional mismatch between the activation space and PE array. However, such small utilization spaces are



(a) PE utilization heatmaps of 2-D mesh PE array with a fixed starting point



(b) PE utilization heatmaps of torus-connected PE array after wear-leveling

Fig. 3. (a) PE utilization heatmaps of Eyeriss [2] with a fixed starting point show severe PE usage imbalance. (b) The torus-connected PE array enables rotating utilization spaces and achieves balanced PE usage.

rare, as optimized scheduling aims to increase PE utilization ratios. In contrast, utilization spaces 2 and 3 cover more than a quarter of the PE array. Even if they are shifted from side to side in the 2-D mesh PE array, PEs at the center cannot avoid overlaps, which only relocates the stress hotspot from the lower-left corner to the center of the PE array.

IV. Methodology

A. Rotational Torus Accelerator

To overcome the limitations of 2-D mesh-style PE arrays for reliability, this paper presents a *rotational torus accelerator (RoTA)*, as shown in Fig. 1. Torus connections are highlighted in red in addition to the existing global and local networks of a conventional accelerator design. In the torus topology, every row or column of PEs forms a unidirectional ring. The loop-around connections in the figure may give an illusion that critical paths are made with long wires, but it is well known that a torus can be easily implemented without such long wires by interconnecting every other PE. More discussions are in Section V-D. The torus-based PE array enables utilization spaces to rotate around the edges so that the starting point of data tile computations can be located anywhere in the PE array, as shown in the leftmost plot of Fig. 3b.

Fig. 4 shows how the torus-based PE array enables wear-leveling. Since every ring of PEs makes a loop, the torus can be unfolded as if it had virtually infinite connections. The figure shows an example of shifting utilization spaces over the unfolded PE arrays. The first data tile exercises PEs in utilization space 1, and the second tile uses the next utilization space to the right. If a utilization space reaches the edge of the PE array (e.g., utilization space U-1), the torus enables crossing the boundary, which technically rotates around the PE array. In utilization space U-1, a pair of PEs such as (w,1) and (1,1) are logically distant but physically adjacent. Although Fig. 4 shows only the case of horizontally striding utilization spaces, the same concept applies to vertical strides.

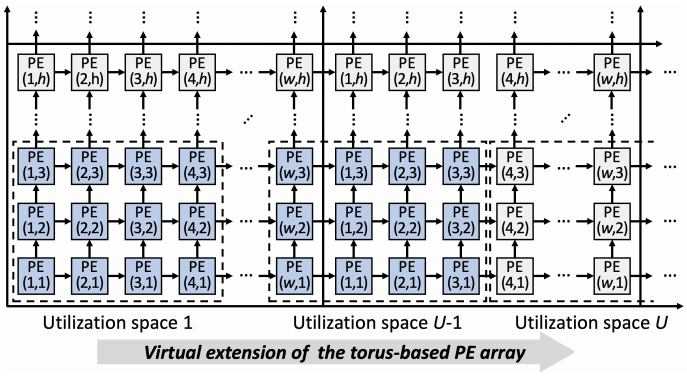


Fig. 4. Horizontally and vertically unfolded torus-based PE array with striding utilization spaces as if it had infinite connections.

B. Lifetime Reliability of PE Array

We use a Weibull distribution to evaluate the relative lifetime reliability of a torus-based PE array with a wear-leveling solution compared to a conventional 2-D mesh PE array using static utilization spaces as the baseline. Eq. (1) shows the reliability function $R(t)$ of the Weibull distribution, representing a single PE. In the equation, $F(t)$ is the cumulative distribution function (CDF), and t is the PE's stress time. β is a shape parameter set to 3.4 based on JEDEC standards [10]. η is a scaling parameter treated as a constant when comparing the relative lifetime reliability.

$$R_{\text{Weibull}}(t) = 1 - F_{\text{Weibull}}(t) = e^{-(\frac{t}{\eta})^{\beta}} \tag{1}$$

The accelerator is operable only when all PEs survive. The reliability function of the PE array is expressed as the serial chain of all PEs, which is the product of individual PE's reliability functions, as shown in Eq. (2). $\alpha_{i,j}$ is the relative active duration of a PE at the (i, j) location in the PE array.

$$R_{\text{Array}}(t) = \prod_{i,j} R_{i,j}(t \times \alpha_{i,j}) = e^{\sum_{i,j} -(\frac{t}{\eta} \times \alpha_{i,j})^{\beta}} \tag{2}$$

Since the mean of a generic Weibull distribution is known as $\eta\Gamma(1+1/\beta)$, the mean time to failure (MTTF) of the whole PE array is calculated as Eq. (3).

$$\text{MTTF}_{\text{Array}} = \frac{\eta}{(\sum_{i,j}(\alpha_{i,j})^{\beta})^{\frac{1}{\beta}}}\Gamma(1+\frac{1}{\beta}) \tag{3}$$

Thus, the relative lifetime improvement of a wear-leveling solution (WL) over the baseline (B) is calculated as Eq. (4), where $\alpha^{\text{WL}}_{i,j}$ and $\alpha^{\text{B}}_{i,j}$ are the relative active time coefficients of a PE for the wear-leveling and baseline cases, respectively.

$$\text{Reliability improvement} = \frac{(\sum_{i,j}(\alpha^{\text{B}}_{i,j})^{\beta})^{\frac{1}{\beta}}}{(\sum_{i,j}(\alpha^{\text{WL}}_{i,j})^{\beta})^{\frac{1}{\beta}}} \tag{4}$$

C. Rotational Wear-Leveling

To process a large neural layer on an accelerator, it is split into many smaller tiles. Table I summarizes the notations of the *rotational wear-leveling (RWL)* scheme. The first data tile exercises a utilization space at the lower-left corner of the PE array, labeled as the 1st scheduling in Fig. 5. To activate the unused portion of the PE array, the next utilization space is shifted horizontally, shown as the 2nd scheduling. Since the PE array ($w \times h$) and utilization space ($x \times y$) dimensions do not match in most cases, leveling the PE usage requires X times of horizontal strides, defined as Eq. (5). To shift utilization spaces X times, the torus-based PE array needs to be unfolded W times horizontally, as shown in Eq. (6).

TABLE I
NOTATIONS OF WEAR-LEVELING ALGORITHM

Parameters	Descriptions
w	# of PEs in the horizontal direction of the PE array
h	# of PEs in the vertical direction of the PE array
x	# of row-wise PEs in a utilization space (i.e., width)
y	# of column-wise PEs in a utilization space (i.e., height)
W	# of PE array unfolding in the horizontal direction
H	# of PE array unfolding in the vertical direction
X	# of utilization space strides in the horizontal direction
Y	# of utilization space strides in the vertical direction
Z	Total # of data tiles (i.e., # of utilization spaces)
D_{max}	Max usage difference between PEs
A_{PE}	# of utilization space allocations per PE (i.e., usage)
R_{diff}	The ratio of D_{max} over min(A_{PE})

$$X = \text{LCM}(w, x)/x \tag{5}$$

$$W = \text{LCM}(w, x)/w \tag{6}$$

Fig. 5 is shown with the C5 layer of ResNet [6] using 8×8 utilization spaces for Z = 32 tiles on the 14×12 PE array of Eyeriss [2]. For w = 14 and x = 8, their least common multiple (LCM) is 56, with X = 7 and W = 4. Thus, utilization spaces are shifted seven times horizontally, and the torus-based PE array is unfolded four times. Once the PE array is horizontally wear-leveled, the (X+1)th utilization space moves vertically and repeats the same process of horizontal striding, as shown in Fig. 5. The number of vertical strides for horizontally wear-leveled utilization spaces is the total tile count (Z) divided by the number of horizontal strides (X), as Eq. (7). The ResNet C5 layer gets Y = 4, meaning that utilization spaces complete four rows of horizontal striding.

$$Y = \lfloor Z/X \rfloor \tag{7}$$

Shifting utilization spaces around the PE array leaves a few residues. To address this, the rightmost case of Fig. 5 is split into two parts. The PE array is perfectly wear-leveled in the bottom part (i.e., usage diff = 0) but partially in the upper part. In the bottom part, the number of PE array unfoldings in the vertical direction can be derived from Y and the relative height ratio of the utilization space (y) and PE array (h), as Eq. (8). For Y = 4, y = 8, and h = 12 in Fig. 5, H_{RWL} = 2 means that the PE array is vertically wear-leveled twice.

$$H_{\text{RWL}} = \lfloor Y \times y/h \rfloor \tag{8}$$

In the upper part of the rightmost case in Fig. 5, the PE array is partially wear-leveled. The blue-colored part, which spans two PE arrays horizontally, is fully wear-leveled. However, the striped part uses only a part of the unfolded PE arrays, causing a usage imbalance. Since utilization spaces stride from the lower left to the upper right, the max usage difference occurs between the lower-left and upper-right PEs. The max difference between horizontally unfolded PE arrays is W, and it can be at most

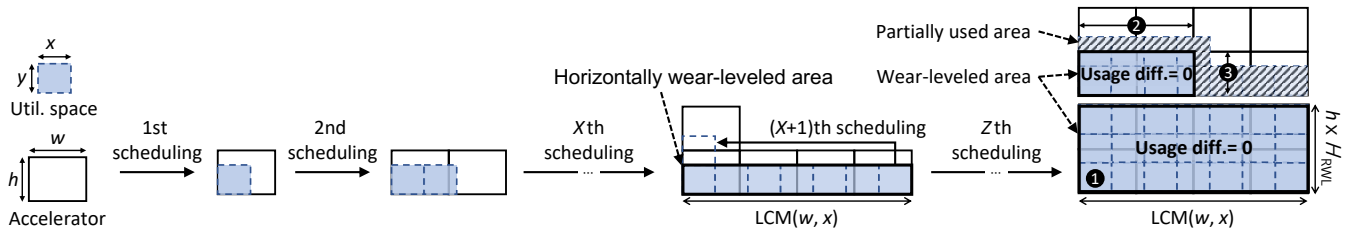


Fig. 5. The RWL scheme shifts utilization spaces horizontally and vertically around the torus-based PE array. The PE array becomes perfectly wear-leveled except for a few data tiles at the end of a layer due to a dimensional mismatch, but it is mathematically proved that the residual usage imbalance is negligible.

Algorithm 1: Wear-Leveling Procedure

```
Parameters: (u, v): starting point of util space
                 L: # of layers in neural network
                 Z: # of data tiles (or util spaces)
   /* Initial starting point of util space */
1: (u, v) ← (1, 1)
   /* Residue-aware optimization */
2: for each l in L; do
     /* Rotational wear-leveling */
3:    for z in Z; do
     /* Process with current utilization space */
4:      process_data_tile();
       /* Horizontal striding of util space */
5:      u ← (u + x - 1) % w + 1
       /* Vertical striding of util space */
6:      if u == 1; then
7:        v ← (v + y - 1) % h + 1
8:      end if
9:    end for
10: end for
```

1 between vertically unfolded ones. In sum, the max usage difference between PEs is given as Eq. (9).

$$D_{\max} \leq W + 1 \tag{9}$$

If the max PE usage difference is much smaller than the smallest PE usage count, PEs have similar usage counts with negligible differences. The min PE usage count for executing a neural layer is found as Eq. (10). ❶ corresponds to the number of unfolded PE arrays in the bottom part of the rightmost step in Fig. 5, which is perfectly wear-leveled. ❷ is the width of the wear-leveled region in the upper part (i.e., blue-colored box), and ❸ is the height of it in the unit of PE arrays.

$$\min(A_{\text{PE}}) = \underbrace{(W \times H_{\text{RWL}})}_{❶} + \underbrace{(\lfloor (Z \ \% \ X) \times x/w \rfloor)}_{❷} \times \underbrace{(\lfloor \lceil Z/X \rceil \times y/h \rfloor - H_{\text{RWL}})}_{❸} \tag{10}$$

The ratio of the max PE usage difference over the min usage count is given as Eq. (11). Since $D_{\max}$ is much smaller than $\min(A_{\text{PE}})$ in general, R_{diff} approaches 0. For example, ResNet [6] has $R_{\text{diff}} = 0.01$ for $D_{\max} = 1.76$ and $\min(A_{\text{PE}}) = 170.4$, meaning that PEs have only 1% peak-to-peak usage difference. As such, RWL effectively balances the PE utilization.

$$R_{\text{diff}} = D_{\max}/\min(A_{\text{PE}}) \simeq 0 \tag{11}$$

D. Residual Optimization

RWL effectively balances the PE usage but leaves a residual imbalance at the end of a layer. The max usage difference between PEs is small per layer, but resetting the starting point of every layer makes the residual imbalance accumulate over time. Thus, RWL should not be applied independently to each layer but relayed across layers. *Residual optimization (RO)* extends RWL such that utilization spaces are shifted around the torus-based PE array across neural layers and networks. The key is that the starting point of a layer's first utilization space must continue from the last utilization space of the preceding layer. Since every layer has different-sized utilization spaces, RO makes some PEs unused or duplicatedly used when transitioning between layers. However, circulating utilization spaces around torus-connected PEs effectively disperses such a gap across the entire PE array in an unbiased fashion.

E. Wear-Leveling Procedure

Algorithm 1 shows the wear-leveling procedure of the RWL +RO policy. The initial starting point of a utilization space is set to the lower-left corner of the PE array as usual (line 1). After processing a data tile, the next utilization space strides horizontally (lines 4-5) based on the PE array width (w) and the utilization space width (x). If the horizontal coordinate (u) of the next utilization space loops back to the leftmost PE, it implies that the LCM of the utilization space width and the PE array width has reached (line 6). Then, the vertical coordinate (v) of the next utilization is shifted (line 7). The loop continues across layers without resetting the starting point of utilization spaces for residual optimization (line 2).

F. Wear Leveling Implementation in Accelerator Controller

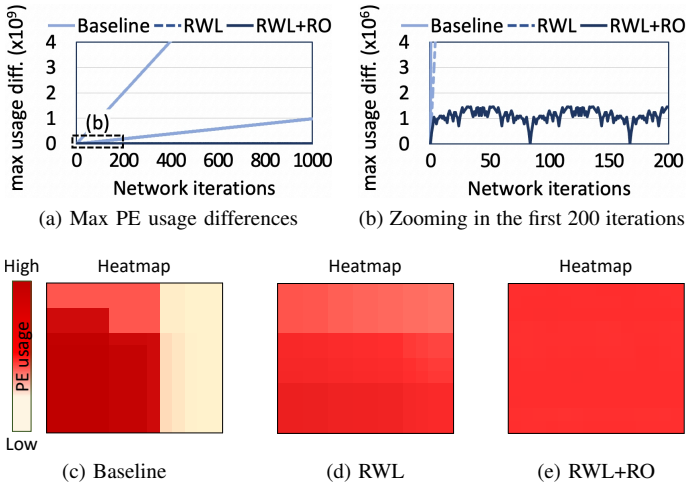
The RWL+RO scheme can be handily implemented in the mapping controller of a conventional accelerator design. The wear-leveling parameters of Algorithm 1 (i.e., w, h, x, and y) are deterministically identifiable before initiating a layer computation without runtime overhead. The controller only needs to track the (u, v) coordinate as shown in lines 5 and 7 of Algorithm 1, simply using circular counters (e.g., $1 \rightarrow 2 \rightarrow \cdots \rightarrow w \rightarrow 1$). Since every striding utilization space still spans continuously from its starting point, it makes no fundamental changes to how data are scattered to and gathered from PEs, except that a utilization space starts from (u, v) instead of (0, 0). The (u, v) coordinate of the next utilization space is updated during the data tile processing period (line 4) and does not incur a cycle penalty. Thus, the RWL+RO scheme adds little overhead to the existing controller logic of the accelerator.

V. Evaluation

The proposed RWL+RO scheme is evaluated based on an accelerator design shown in Fig. 1, with torus connections for RoTA and a mesh-based PE array for the baseline. The PE

TABLE II
DNN WORKLOADS USED IN EXPERIMENTS

DNN domains	Networks	abbr.	Features
Image classification	ResNet-50 [6]	Res	Residual blocks
	Inception v4 [22]	Inc	Asymmetric weights
Object detection	YOLO v3 [18]	YL	Large dataset
Lightweight network	SqueezeNet [9]	Sqz	Small weights
	MobileNet v3 [7]	Mb	Group Conv.
	EfficientNet [24]	Eff	MBConv. blocks
Transformer	ViT [4]	VT	Transformer encoding
	MobileViT [12]	MVT	Embedded transformer
	Llama v2 [26]	LM	Large language model



(a) Max PE usage differences (b) Zooming in the first 200 iterations

(c) Baseline (d) RWL (e) RWL+RO

Fig. 6. (a) Max PE usage difference of three schemes for 1,000 iterations of SqueezeNet on a 14×12 PE array. (b) The max usage difference of the RWL+RO scheme is bounded, but others grow unboundedly. (c), (d), (e) PE usage heatmaps of baseline, RWL, and RWL+RO after 1,000 iterations.

array size is 14×12, and each PE has 24, 448, and 48-byte local buffers for input, weight, and output, respectively. The size of the shared GLB is 108KB [2]. Table II lists DNN workloads used in our experiments, where diverse shapes of neural layers exercise different-sized utilization spaces. The size of each layer's utilization space is obtained from NeuroSpector [15] for energy-optimal execution, and we composed a simulator to track the usage count of individual PEs,

A. Effectiveness of Wear-Leveling Solution

Fig. 6a compares max PE usage differences between three cases: i) baseline without wear-leveling, ii) RWL-only, and iii) RWL+RO when executing SqueezeNet for 1,000 batches. The baseline always uses utilization spaces anchored at the lower-left corner of the PE array, causing its max PE usage difference to grow rapidly. With RWL, the slope of the max PE usage difference is substantially reduced but still increasing unboundedly since a residual imbalance accumulates in every layer. The result of RWL+RO is technically invisible in Fig. 6a, so Fig. 6b zooms in the first 200 iterations. It reveals that the max PE usage difference of RWL+RO is bounded effectively, proving that the usage imbalance is unbiased to PE locations and non-cumulative over time. Figs. 6c-e plot the resulting PE usage heatmaps of the three cases after 1,000 iterations. The baseline shows severe utilization imbalance in the PE array. RWL alone substantially alleviates the PE usage imbalance but still shows clear divides between PEs. In contrast, RWL+RO achieves nearly perfect wear-leveling.

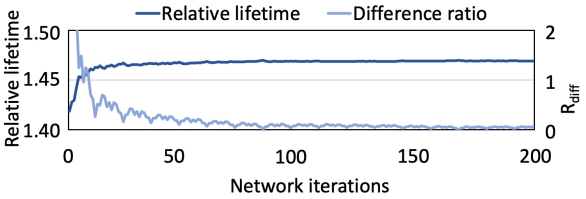


Fig. 7. Projected accelerator lifetime vs. R_{diff}. RWL+RO makes R_{diff} converge to 0, implying that PEs are effectively wear-leveled. The projected lifetime inversely follows the R_{diff} trend.

B. Lifetime Reliability Improvement

Fig. 7 draws the transient progress of the accelerator's projected lifetime and R_{diff} for the first 200 iterations of SqueezeNet using RWL+RO. As the execution progresses, R_{diff} converges to 0, implying that PEs are almost perfectly wear-leveled. The expected accelerator lifetime inversely follows the R_{diff} trend. Fig. 8 shows the relative lifetime improvement of wear-leveling methods estimated based on Eq. (4) for various DNN workloads. On average, the RWL+RO scheme improves the accelerator lifetime by 1.69x over the baseline. Applying the RWL technique alone without residual optimization gives comparable lifetime improvements, earning an average 1.65x increase over the baseline. However, a few DNN workloads exhibit noticeable gaps between the RWL-only and RWL+RO methods, such as MobileNet v3, EfficientNet, and MobileViT. For these DNNs, the RWL+RO scheme shows a 1.55x lifetime improvement, while the RWL-only method earns a 1.46x gain on average. These DNN workloads are relatively small in size compared to others, so they have less opportunity to exploit the RWL effect within each small layer. As a result, the RWL-only method experiences the gradual accumulation of residual PE usage imbalance over time. In general, lifetime improvements exhibit a strong correlation with PE utilization ratios shown in Fig. 2a. For instance, YOLO v3 [18] layers have the lowest PE utilization ratios among the tested DNN workloads but show the greatest lifetime increase of 2.37x.

C. PE Utilization vs. Reliability Improvement

To evaluate how close the proposed wear-leveling method approaches the theoretically achievable upper bound of reliability enhancement, Fig. 9 plots the lifetime improvement of individual neural layers with respect to their PE utilization ratios. In the baseline without wear-leveling, PEs participating in the layer computation have the active time coefficient of α = 1, and other inactive PEs have $\alpha = 0$. Applying these to Eq. (4), the baseline case in the numerator of the equation becomes $(x \times y)^{1/\beta}$. On the contrary, the perfect wear-leveling makes all PEs have the same active time coefficient of $(x \times y)/(w \times h)$, and the wear-leveling case in the denominator of Eq. (4) can be simplified as $(x \times y) \times (w \times h)^{1/\beta - 1}$. Thus, the upper bound of the lifetime improvement with the perfect wear-leveling is $[(w \times h)/(x \times y)]^{1/\beta - 1}$. Since $(x \times y)/(w \times h)$ represents the utilization ratio of the PE array, the theoretical upper bound of the lifetime reliability enhancement becomes $(\text{PE utilization})^{1/\beta - 1}$.

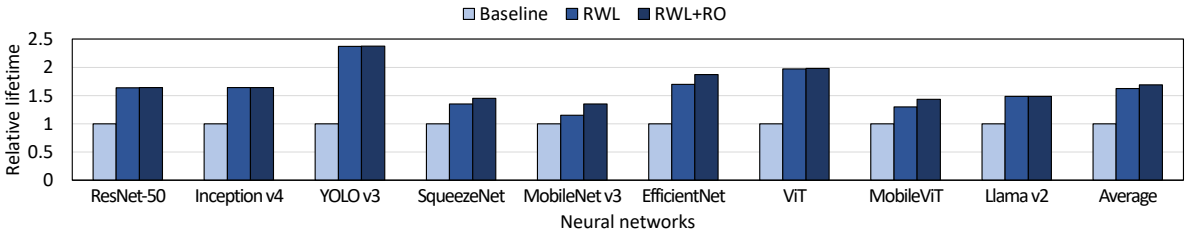


Fig. 8. Comparison of relative lifetime between the baseline, RWL-only, and RWL+RO schemes for various DNN workloads. The RWL+RO scheme enhances the accelerator lifetime by 1.69x on average over the baseline. The RWL-only method shows inefficiency with MobileNet, EfficientNet, and MobileViT.

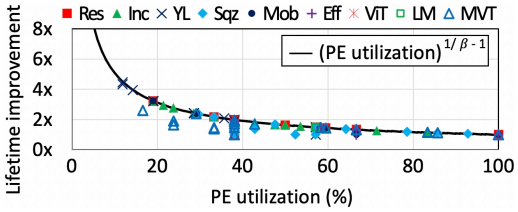


Fig. 9. Layer-wise lifetime improvements with respect to PE utilization ratios for various DNNs. RWL closely approaches the theoretically achievable upper bound of reliability enhancement.

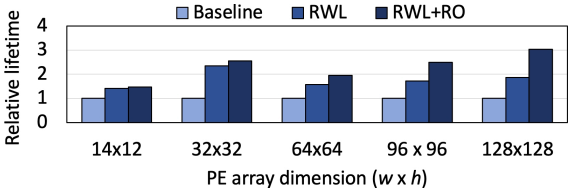


Fig. 10. The lifetime improvement of wear-leveling schemes for increasing the PE array size with SqueezeNet. The RWL+RO scheme achieves greater lifetime reliability enhancements for larger PE arrays.

Fig. 9 shows that the results of per-layer RWL closely approach the theoretical upper bound. Several layers have lower improvements than the upper bound, but their gaps can be effectively mitigated by engaging residual optimization with RWL.

D. Design and Scheduling Overhead

Adding torus connections to the local network (i.e., inter-PE wires) of a conventional 2-D mesh PE array has a negligibly small design overhead. It is well known that a torus can be easily implemented without long loop-back wires, illustrated in Fig. 1, by interconnecting every other PE in a zigzag fashion. Since the existing local network contributes to a small portion of the overall accelerator area (mostly taken by buffers and logic), adding another short wire to each PE row and column adds a very small overhead. We synthesized the RoTA structure using the Synopsis Design Compiler based on the SAED 32nm technology. The result tells us that the torus-connected PE array has only 0.3% design overhead compared to the conventional 2-D mesh PE array.

In addition, wear-leveling logic can be easily implemented using four registers to hold the w, h, x, and y values in Algorithm 1 and two circular counters to track the (u, v) coordinate. The wear-leveling parameters are deterministically identifiable before initiating a layer computation without runtime overhead. Likewise, the (u, v) coordinate of the next utilization space can be updated by incrementing the circular counters during the data tile processing period without a cycle penalty. Thus, the RoTA design with the RWL+RO scheme can be implemented with little design overhead and no performance degradation compared to the baseline.

E. Wear-Leveling of Large PE Arrays

To demonstrate that the wear-leveling scheme is not limited to a specific PE array size, Fig. 10 compares the relative lifetime of the baseline, RWL-only, and RWL+RO schemes on different-sized PE arrays. The graph shows that the RWL+RO scheme achieves greater lifetime reliability enhancements with larger PE arrays. Since increasing the PE array size tends to diminish the PE utilization ratio and leave a larger residual imbalance, it adversely impacts the lifetime reliability. This opens up a bigger opportunity for the RWL+RO scheme. As a result, the proposed wear-leveling solution achieves greater lifetime reliability improvements for larger PE arrays.

VI. Related Work

System-level lifetime reliability issues were widely explored in computer systems, including processors [11], [21], cache and memory [27], [30], and storage devices [3], [28]. However, the reliability issues have not been much explored for neural accelerator designs and operations. A few studies explored the vulnerability of bit flips due to low-voltage operations in DNN accelerators and their consequential impacts on inference accuracy, such as protecting critical data bits [17], [29], [13], leveraging error-resistant memory [1], [5], etc. However, these efforts dealing with transient errors are orthogonal to the lifetime reliability problem addressed in this paper.

Torus networks were traditionally used in telecommunications, interconnection networks, etc. Microsoft Catapult [16] is known to use a torus topology for connecting multiple FPGA boards for high-throughput communications. However, torus designs were never introduced to interconnect on-chip PEs for wear-leveling in the related literature.

VII. Conclusion

This paper presented a reliability-aware design for neural accelerators and a wear-leveling solution to eliminate usage imbalance between PEs. The proposed RoTA design enables circulating PE utilization spaces via torus connections, and the simple implementation of the RWL+RO policy effectively balances PE utilization. Experimental results show that RoTA with RWL+RO improves lifetime reliability by 1.69x with only 0.3% design overhead and no performance degradation.

REFERENCES

[1] A. Azizimazreah, Y. Gu, X. Gu, and L. Chen, "Tolerating Soft Errors in Deep Learning Accelerators with Reliable On-Chip Memory Designs," *IEEE International Conference on Networking, Architecture and Storage*, Oct. 2018, pp. 1–10.

[2] Y. Chen, T. Krishna, J. Emer, and V. sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, pp. 127–138, Jan. 2017.

[3] Dharamjeet, Y. Chen, T. Chen, Y. Kuan, and Y. Chang, "LLSM: A Lifetime-Aware Wear-Leveling for LSM-Tree on NAND Flash Memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3946–3956, Aug. 2022.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," pp. 1–22, Oct. 2020, [Online], Available: https://arxiv.org/abs/2010.11929.

[5] S. Hari, M. Sullivan, T. Tsai, and S. Keckler, "Making Convolutions Resilient via Algorithm-Based Error Detection Techniques," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2546–2558, July 2022.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2016, pp. 770–778.

[7] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. Le, and H. Adam, "Searching for MobileNetV3," *IEEE/CVF International Conference on Computer Vision*, Oct. 2019, pp. 1314–1324.

[8] Q. Huang, M. Kang, G. Dinh, T. Norell, A. Kalaiah, J. Demmel, J. Wawrzynek, and Y. Shao, "CoSA: Scheduling by Constrained Optimization for Spatial Accelerators," *ACM/IEEE International Symposium on Computer Architecture*, June 2021, pp. 554–566.

[9] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer, "SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size," pp. 1–13, Feb. 2016, [Online], Available: https://arxiv.org/abs/1602.07360.

[10] JEDEC Publication, "Failure Mechanisms and Models for Semiconductor Devices," JEDEC Solid State Technology Association, JEP122H, Sept. 2016.

[11] H. Kim, A. Vitkovskiy, P. Gratz, and V. Soteriou, "Use It or Lose It: Wear-Out and Lifetime in Future Chip Multiprocessors," *IEEE/ACM International Symposium on Microarchitecture*, Dec. 2013, pp. 136–147.

[12] S. Mehta and M. Rastegari, "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer," pp. 1–26, Oct. 2021, [Online], Available: https://arxiv.org/abs/2110.02178.

[13] E. Ozen and A. Orailoglu, "Just Say Zero: Containing Critical Bit-Error Propagation in Deep Neural Networks with Anomalous Feature Suppression," *IEEE/ACM International Conference on Computer Aided Design*, Nov. 2020, pp. 1–9.

[14] A. Parashar, P. Raina, Y. Shao, Y. Chen, V. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. Keckler, and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," *IEEE International Symposium on Performance Analysis of Systems and Software*, Mar. 2019, pp. 304–315.

[15] C. Park, B. Kim, S. Ryu, and W. Song, "NeoroSpector: Systematic Optimization of Dataflow Scheduling in DNN Accelerator," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 8, pp. 2279–2294, June 2023.

[16] A. Putnam, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Xiao, and D. Burger, "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," *IEEE/ACM International Symposium on Computer Architecture*, June 2014, pp. 13–24.

[17] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. Lee, J. Hernández-Lobato, G. Wei, and D. Brooks, "Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators," *IEEE/ACM International Symposium on Computer Architecture*, June 2016, pp. 267–278.

[18] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," pp. 1–6, Apr. 2018, [Online], Available: https://arxiv.org/abs/1804.02767.

[19] A. Samajdar, E. Qin, M. Pellauer, and T. Krishna, "Self Adaptive Reconfigurable Arrays (SARA): Learning Flexible GEMM Accelerator Configuration and Mapping-space using ML," *ACM/IEEE Design Automation Conference*, July 2022, pp. 583–588.

[20] Y. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. Tell, Y. Zhang, W. Dally, J. Emer, C. Gray, B. Khailany, and S. Keckler, "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture," *IEEE/ACM International Symposium on Microarchitecture*, Oct. 2019, pp. 14–27.

[21] W. Song, S. Mukhopadhyay, and S. Yalamanchili, "Amdahl's Law for Lifetime Reliability Scaling in Heterogeneous Multicore Processors," *IEEE International Symposium on High-Performance Computer Architecture*, Mar. 2016, pp. 594–605.

[22] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *AAAI Conference on Artificial Intelligence*, Feb. 2017, pp. 4278–4284.

[23] E. Talpes, D. Sarma, G. Venkataramanan, P. Bannon, B. McGee, B. Floering, A. Jalote, C. Hsiong, S. Arora, A. Gorti, and G. Sachdev, "Compute Solution for Tesla's Full Self-Driving Computer," *IEEE Micro*, vol. 40, no. 2, pp. 25–35, Mar. 2020.

[24] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *International Conference on Machine Learning*, June 2019, pp. 6105–6114.

[25] J. Tong, A. Itagi, P. Chatarasi, and T. Krishna, "FEATHER: A Reconfigurable Accelerator with Data Reordering Support for Low-Cost On-Chip Dataflow Switching," *ACM/IEEE International Symposium on Computer Architecture*, June 2024, pp. 198–214.

[26] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. Smith, R. Subramanian, X. Tan, B. Tang, R. Taylor, A. Williams, J. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open Foundation and Fine-Tuned Chat Models," pp. 1–77, July 2023, [Online], Available: https://arxiv.org/abs/2307.09288.

[27] J. Wang, X. Dong, Y. Xie, and N. Jouppi, "i2WAP: Improving Non-volatile Cache Lifetime by Reducing Inter- and Intra-set Write Variations," *IEEE International Symposium on High-Performance Computer Architecture*, Feb. 2013, pp. 234–245.

[28] S. Wang, F. Wu, C. Yang, J. Zhou, C. Xie, and J. Wan, "WAS: Wear Aware Superblock Management for Prolonging SSD Lifetime," *ACM/IEEE Design Automation Conference*, June 2019, pp. 1–6.

[29] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: Enabling Aggressive Voltage Underscaling and Timing Error Resilience for Energy Efficient Deep Learning Accelerators," *ACM/IEEE Design Automation Conference*, June 2018, pp. 1–6.

[30] J. Zhang, C. Wang, Z. Zhu, D. Kline, A. Jones, H. Yang, and Y. Wang, "Realizing Extreme Endurance Through Fault-aware Wear Leveling and Improved Tolerance," *IEEE International Symposium on High-Performance Computer Architecture*, Feb. 2023, pp. 964–976.