Reliability-Aware Quantization for Anti-Aging NPUs

Sami Salamin*, Georgios Zervakis*, Ourania Spantidi†,
Iraklis Anagnostopoulos†, Jörg Henkel* and Hussam Amrouch‡*
*Chair for Embedded Systems (CES), Karlsruhe Institute of Technology, Karlsruhe, Germany
†Department of Electrical and Computer Engineering, Southern Illinois University, Carbondale, U.S.A.
‡Chair of Semiconductor Test and Reliability (STAR), University of Stuttgart, Stuttgart, Germany
*{sami.salamin, georgios.zervakis, henkel}@kit.edu, †{ourania.spantidi, iraklis.anagno}@siu.edu, ‡amrouch@iti.uni-stuttgart.de

***Abstract*—Transistor aging is one of the major concerns that challenges designers in advanced technologies. It profoundly degrades the reliability of circuits during its lifetime as it slows down transistors resulting in errors due to timing violations unless large guardbands are included, which leads to considerable performance losses. When it comes to Neural Processing Units (NPUs), where increasing the inference speed is the primary goal, such performance losses cannot be tolerated. In this work, we are the first to propose a reliability-aware quantization to eliminate aging effects in NPUs while completely removing guardbands. Our technique delivers a graceful inference accuracy degradation over time while compensating for the aging-induced delay increase of the NPU. Our evaluation, over ten state-of-the-art neural network architectures trained on the ImageNet dataset, demonstrates that for an entire lifetime of 10 years, the average accuracy loss is merely 3%. In the meantime, our technique achieves 23% higher performance due to the elimination of the aging guardband.**

***Index Terms*—Approximate Computing, Adaptive Approximation, Aging, Neural Networks, Quantization, Reliability**

I. Introduction

Late advancements in Neural Networks (NNs) research boosted the accuracy of several machine learning applications, at the cost of an immense increase in computational demands [1]. However, to achieve that and to bring the inference speed of Deep NNs (DNNs) to an acceptable level, custom ASIC Neural Processing Units (NPUs) are becoming ubiquitous in general purpose and embedded computing [1]–[3]. An NPU consists of thousands of multiply-accumulate (MAC) units [4], which provide massive parallelism of the performed computations that DNNs demand. Google TPU integrates 64K MACs [1], while even the embedded-oriented Samsung [3] and Google [2] NPUs employ 1K and 4K MACs, respectively. In NPUs, a large number of MAC units are tightly packed together within a small footprint. Their inherent nature to simultaneously perform tens of tera operations per second, makes NPUs subject to elevated on-chip power densities that rapidly result in excessive on-chip temperatures during operation [4].

Circuit Aging: The very high utilization of MAC circuits within NPUs [1], [4] exposes the underlying transistors to continuous stress with very little time for relaxation and recovery. As a result, transistors age faster. In addition, the presence of excessive temperatures, as mentioned earlier, exacerbates further the problem as the majority of mechanisms behind transistor aging exponentially depend on the operating temperature [5], [6]. Generated defects, due to transistor aging phenomena such as Bias Temperature Instability (BTI) and Hot-Carrier Injection (HCI), manifest themselves as a degradation in the main electrical characteristics of transistors. In practice, the threshold voltage (V_{th}) increases [6] leading, in turn, to a reduction of the drain current of a transistor in the ON state (I_{on}). This considerably increases the propagation delay of the transistor and thus, of the logic cells (Eq. 1-2 [7]). Hence, circuits exhibit timing errors because the operating frequency becomes unsustainable over time. To overcome that and keep aging effects at bay for the entire projected lifetime (e.g., 10 years), a timing guardband (t_{GB}) must be included on top of the critical path delay (Eq. 3) at *design time*. This leads directly to large losses in performance from the beginning until the end of lifetime even through aging-induced delay degradations do not yet exist (or are very small) at the early phases of the chip's lifetime. Hence, the cost of guardbanding is paid from the very beginning even though it is not yet needed (Eq. 4). Several approaches have been proposed [8], [9] w.r.t. guardband narrowing and aimed at minimizing the associated performance losses. However, they reduce the aging impact by inducing area/power overhead, through transistor over-sizing [8].

$$I_{on} \propto V_{dd} - (V_{th} + \Delta V_{th}) \tag{1}$$

$$t_{CP}(fresh) = \sum_{m_i \in CP} D_{m_i};\ D_{m_i} \approx \frac{CV_{dd}}{4}\left(\frac{1}{I_{onN}} + \frac{1}{I_{onP}}\right) \tag{2}$$

$$t_{freq}(x) < t_{CP}(fresh) + t_{GB}(y),\ x \leqslant y \Rightarrow \textit{timing errors}\ \textbf{!} \tag{3}$$

$$t_{freq}(x) > t_{CP}(fresh) + t_{GB}(y),\ x < y \Rightarrow \textit{Perf. loss}\ \textbf{!} \tag{4}$$

where x is chip's age, x=0 as fresh chip. y is projected lifetime. m_i refers to transistors that form the circuit's critical path (CP). D_{m_i} is simplified propagation delay of logic gate [7], and C represents the load capacitance connected to the gate.

Aging-Aware Approximation: Recently, approximate computing has been employed to address aging in error-tolerant applications [10], [11]. Approximate computing exploits the inherent error resilience of several applications, to trade-off computational accuracy with other metrics, e.g., delay [10]–[14]. Aging-aware works in approximate computing introduce directed approximations to improve a circuit's performance and mitigate the aging effects. However, they examined very simple topologies, e.g., RCA adders and array multipliers [10], [11].

In this work, we suppress aging effects in NPUs by applying, for the first time, adaptive approximation through input compression in which reliability-aware quantization is used. With a marginal inference accuracy loss, we demonstrate that aging guardbands can be removed for the entire projected lifetime.

Our novel contributions within this paper are as follows:

(1) This is the first work that employs quantization as a novel mechanism to eliminate aging effects in NPUs.

(2) We present, for the first time, a graceful-approximation technique that suppresses, over time, aging effects in NPUs. Our technique enables designers to remove aging guardbands and hence eliminates the associated performance loss.

(3) We demonstrate that for an average accuracy loss of merely 3%, our technique eliminates the performance loss due to aging

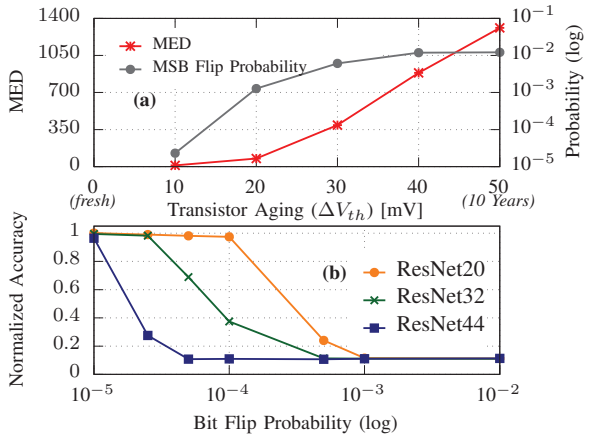

Fig. 1: a) The error characteristics of an 8-bit multiplier under aging. b) Accuracy of three ResNets when injecting errors (random bit flips in the 2 MSBs with a given probability) at the multiplications of the convolutional layers.

guardband, i.e., 23%, while timing errors caused by transistor aging are suppressed for the entire lifetime of 10 years.

II. Related Work

In [12]–[14] approximate multipliers are employed and run-time reconfigurable approximate NN inference accelerators are implemented. However, [12]–[14] target power and not delay optimization. In [11], fixed approximation through precision scaling is applied in order to narrow or remove aging guard-bands. Nevertheless, fixed approximation leads to constant quality degradation that cannot be adapted over time. In [10], adaptive input cutting and masking techniques are proposed to mitigate aging and achieve a graceful accuracy degradation of a DCT/IDCT accelerator. However, [10] was only applied to the very slow ripple-carry adder and array multiplier. Moreover, [10] requires control circuitry to set the run-time approximation. [10], [11] reduce the computational precision of the accelerator itself. Hence, considering that errors due to approximate hardware are input-dependent, by just omitting some bits from the computations [10], [11], the quality loss for some inputs might be unacceptable [12].

III. Aging-Induced Timing Errors

Aging-induced timing degradation increases the circuit's delay over time and thus, timing errors occur when the circuit's paths fail to fulfill the required timing constraints [11] (Eq. 3). Induced timing errors result in unacceptable accuracy loss even after just 1 year, as shown in [10]. In arithmetic circuits, errors mainly occur in the most significant bits (MSBs) [10]. To explore that, we demonstrate in Fig. 1a, the aging-induced timing errors of a 8-bit multiplier circuit (i.e., the vital component of any NPU). The circuit is obtained from the commercial Synopsys DesignWare library. Fig. 1a presents i) the Mean Error Distance (MED) for different aging levels (represented as different values in threshold voltage increase (ΔV_{th})) and ii) the resulting probability of a bit flip in one of the two MSBs. One million random generated inputs are used. MED is defined as the average absolute distance between the output of the accurate and of the aged circuit. ΔV_{th}=50mV represents the end of lifetime, i.e., 10 years [15]. *No timing guardbands are used in this investigation. Hence, the multiplier is clocked with the maximum frequency obtained from operation at the critical path delay of the fresh multiplier*. As shown in Fig. 1a, more errors are produced over time (higher MED), and the probability of a bit-flip at the MSBs increases significantly.

Recent research demonstrated that the deeper NNs become, the more susceptible to errors generated in the multiplier units their accuracy becomes [13]. In Fig. 1b, we present an estimation of how aging-induced timing errors impact the accuracy of different NNs. Note, post-synthesis timing simulation to capture the accuracy loss due to aging is infeasible. DNNs feature millions of multiplications [13], [14] and simulating only one inference requires several hours [13]. Hence, we run inference at software level and we inject errors in the performed multiplications. Error injection is implemented by randomly flipping one of the two MSBs with a given probability, where each experiment is performed ten times to capture the average accuracy. Three ResNets are considered and the probability of the bit flip ranges from 10^{-5} to 10^{-2}. As shown in Fig. 1b, as the probability of a bit flip increases, the accuracy drops significantly and becomes unacceptable after a very small probability level of 5×10^{-4}. In addition, for the deeper NNs (i.e., as the size of ResNet increases) the accuracy drops faster. It is noteworthy that the bit flip probability (~10^{-3}) of the 8-bit multiplier with only a $20mV$ voltage threshold increase due to aging, results in an unacceptable accuracy drop for all the examined NNs. In Fig. 1, for the sake of exemplification, we examine only the bit-flip probability at the MSBs. As shown in Figs. 1, aging-induced timing errors are critical for the NPU's accuracy, leading to poor quality even after a small degradation.

IV. MAC Delay Analysis Under Input Compression

The core component of NPUs is the MAC unit. NPUs perform millions of MAC operations per inference and the overall performance of the NPU is defined by the speed of single MAC unit. In our work, we consider a microarchitecture based on Google Edge TPU [2] that comprises a 64×64 systolic MAC array. Hence, we design a MAC unit that consists of an 8-bit unsigned multiplier and a 22-bit unsigned adder to prevent accumulation overflow. Then, we synthesize the RTL description of the MAC using Synopsys Design Compiler targeting maximum performance using the 14nm FinFET technology which is calibrated with measurement data from Intel. The optimized arithmetic modules of the DesignWare library are used to describe the MAC unit, as done in commercial chips.

Not all timing paths in a circuit feature the same delay and therefore, the paths that will be *activated* are input-dependent. For arithmetic circuits (such as a MAC unit), lower bit-width numbers lead to shorter carry propagations and thus, faster operation [10], [11]. *Exploiting the latter, we compress the inputs of the MAC to achieve faster operation*. The accurate MAC, in practice, performs the operation: $A \times B + C$. The width of A and B is 8 bits while the width of C is 22 bits. The compressed inputs A', B', and C' feature a bit-width of $8-\alpha$, $8-\beta$, and $22-(\alpha+\beta)$, respectively. Since the compressed inputs feature smaller bit-width, we apply zero-padding to the remaining bits. Two options are available for zero-padding: i) MSB-padding, i.e., fill with zeros the MSB positions, and ii)

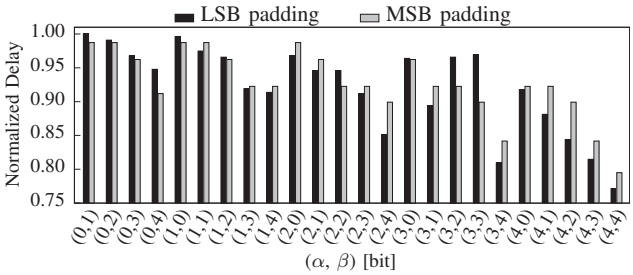

Fig. 2: Delay gain of the 8-bit MAC when applying (α, β) input compression. Both MSB and LSB paddings are evaluated.

LSB-padding, i.e., fill with zeros the LSB positions. In the latter case, the result $A' \times B' + C'$ is shifted left $\alpha + \beta$ places.

In Fig. 2, we evaluate the delay of our MAC unit when compressing its inputs, i.e., performing $A' \times B' + C'$ instead of $A \times B + C$. Various compression values (α, β) and both padding options are examined. As shown in Fig. 2, around 23% delay gain can be achieved for up to $(4, 4)$ compression. In addition, Fig. 2 shows that some compression values are benefited by MSB padding while others by LSB padding. Therefore, both padding options should be considered. Fig. 2 demonstrates that by just compressing the MAC inputs, we can achieve significant delay gain without any circuit modifications.

In order to compress the MAC inputs, while merely impacting the inference accuracy of the NN, we employ multiple low bit-width quantization techniques [16]–[19]. Particularly, we quantize the activations and the weights to $8-\alpha$ and $8-\beta$ bits respectively, and we perform the appropriate padding afterward (Section V). In that way, our approach enables i) accurate operation when no aging effects appear (i.e., no compression), and ii) gradually increase the compression (α and β values) over time to increase the delay gain as the NPU ages.

V. Input Compression Through Quantization

Typically, NNs are trained using a 32-bit floating point (FP32) number representation. Post-training quantization aims at reducing the number of bits used to represent weights and activations of NNs (e.g., 8-bit integers, INT8). Consequently, the model size is reduced while the accuracy is maintained very close to the accuracy of the FP32 model [16].

In this work, we utilize post-training quantization to implement input compression and leverage the respective delay gain proposed in Section IV to address NPU aging. To represent the compressed inputs, A', B', and C', we target quantization with different bits for weights and activations. Therefore, we created a library of multiple low-bit width post-training quantization methods based on recently published approaches. We integrate multiple methods, since some of them are optimized for specific NNs, or they are optimized for very low precision. Some other methods require off-line statistics and in some cases the actual quantization is time-consuming due to the multiple optimizations. Particularly, we have included uniform symmetric [16] and asymmetric min/max quantization [17], as well as more sophisticated methods and libraries such as ACIQ [18] (w/ and w/o bias correction), and LAPQ [19], which support per-channel bit optimizations, stats-based analysis, and bias correction. *All these methods do not require NN retraining and allow the utilization of different precision for weights and activations*. More details about the employed quantization technique in [16]–[19]. As an example to demonstrate that different methods have different effects on NNs, for ResNet50, LAPQ is the best method for W8A4 (INT8 for weights INT4 for activations) where accuracy drops 1.7% on the ImageNet dataset. For W4A4, ACIQ is the best method with a drop of only 4.2% (corresponding drop of LAPQ is 11.3%), verifying its design principle that targets low bit-width quantization. Contrary, VGG13 on ImageNet, LAPQ is the best quantization method both for W8A4 and W4A4 with accuracy drop of only 1% and 4.2% respectively.

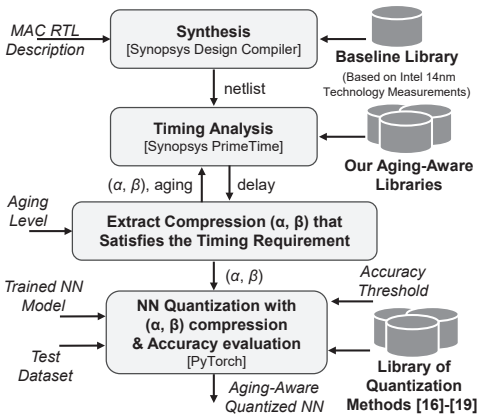

Fig. 3: Our device-to-system implementation that enables aging-aware quantization to suppress aging effects in NPUs.

For 8-bit quantization, the activations and weights are quantized to the $[0,\ 2^8)$ segment, while the biases to $[0,\ 2^{16})$. Considering (α, β) compression, the activations are quantized to $[0,\ 2^{8-\alpha})$, the weights to $[0,\ 2^{8-\beta})$, and the biases to $[0,\ 2^{16-\alpha-\beta)})$. When considering LSB padding, the convolution inputs are shifted left and thus, convolution operation equals:

$$\begin{aligned} F_{shifted} &= Bias \times 2^{(\alpha+\beta)} + \sum_{\forall j} \left((A_j \times 2^{\alpha}) \times (W_j \times 2^{\beta})\right) \\ &= \left(Bias + \sum_{\forall j} \left(A_j \times W_j\right)\right) \times 2^{(\alpha+\beta)} = F \times 2^{(\alpha+\beta)} \end{aligned} \tag{5}$$

Hence, the output needs to be shifted right $\alpha + \beta$ positions. However, the latter does not require additional hardware since it can be easily performed at the software side when storing/reading the convolution result to/from the memory. On the other hand, when MSB padding is used, no shift is required.

VI. Implementation of Aging-Aware Quantization

Here, we describe our implementation (summarized in Fig. 3) to perform the proposed aging-aware quantization. Our implementation starts from the device level all the way up to the system level where the NN inference accuracy is impacted.

A. Aging Modeling

(1) Aging Model: In this work, we employ the state-of-the-art physics-based aging model [20]. This model has been validated against semiconductor measurements for various technologies and transistor structures. It is able to precisely capture aging-induced degradation (ΔV_{th}) for the 14nm technology.

(2) Aging-Aware Cell Libraries: To enable aging support in commercial digital design tool flows, we generate aging-aware cell libraries. First, we calibrate the (BSIM-CMG) model to match Intel's 14nm FinFET technology measurements provided in [21]. Details on calibration and validation of BSIM-CMG are available in [22]. Then, we employ the state-of-the-art physics-based aging model to estimate the corresponding ΔV_{th} over time. In this work, we consider aging from 0 to 10 years for the typical projected lifetime. Aging gradually increases ΔV_{th} over time. ΔV_{th} for a fresh chip is equal to 0, while after 10 years it reaches 50mV, as demonstrated from measurements for the FinFET technology [15], [20]. Aging mechanism is affected by the operating conditions (e.g., utilization and temperature). For instance, $\Delta V_{th} = 20$mV may correspond to 1-2 years. Hence, in our analysis, we consider ΔV_{th} as an unbiased measure of the aging level and we investigate aging effects at different ΔV_{th} levels from 0mV (fresh) to 50mV (10 years) with a step of 10mV. Afterward, for each ΔV_{th} step, we create an aging-aware cell library by characterizing all standard cells on the respective ΔV_{th} based on the open-source FinFET standard cells from Silvaco, using Synopsys SiliconSmart. This is done using the commercial SPICE simulation that measures the delay and power of every standard cell under the influence of ΔV_{th}.
(3) Aging-induced Delay Analysis: Synopsys Design Compiler and the `compile_ultra` command are used to synthesize the circuit's RTL description targeting maximum performance, i.e., zero-slack. For synthesis, we consider the fresh library, i.e., baseline without aging. Next, we use Synopsys PrimeTime to perform static timing analysis (STA) on the post-synthesis netlist. During STA we employ our aging-aware cell libraries to precisely capture the impact of aging on the circuit's delay. We consider the worst-case analysis where all transistors exhibit the maximum degradation. In addition, we analyze the timing information of both the uncompressed and compressed (i.e., reduced bit width) inputs. In the latter case, we specify that the respective input bits of the bit positions that are padded with zeros (due to input compression) are constantly set to '0'. Hence, we precisely obtain the circuit's delay w.r.t. the aging period and the paths that are activated due input compression.

B. Our Proposed Aging-Aware Quantization

In Section IV, we demonstrated that applying input compression on the MAC unit delivers considerable delay gains that can potentially eliminate the aging-induced timing errors. As discussed in Section V, input compression in NPUs can be applied by employing low bit-width quantization. Nevertheless, the latter still results in accuracy loss, despite the numerous quantization methods that have been proposed. In our work, we introduce an adaptive approximation approach by progressively increasing the input compression over time. Our implementation is illustrated in Fig. 3 and described in Algorithm 1. First, we synthesize the RTL description of our circuit as described above (Section VI-A (3)) to obtain the post-synthesis netlist without aging. We consider the MAC unit as our driving circuit since the accuracy and delay of the MAC operation will define the accuracy and speed of the NPU [1], [13]. Next, we use PrimeTime and our aging-aware libraries (Section VI-A (2)) to perform timing analyses to identify all the compression values (α, β), under both MSB and LSB padding, that satisfy the timing constraint of the MAC unit (lines 2-4). Targeting minimum compression, we select the (α, β) that minimizes $\sqrt{\alpha^2+\beta^2}$ (line 5). In the case of a tie, we select the (α, β) with the highest precision for the activations [18] (i.e., smallest α). Finally, we use the obtained compression value (α, β) to quantize the NN model. The quantization size equals $8-\alpha$ for the activations, $8-\beta$ for the weights, and $16-\alpha-\beta$ for the biases. For the quantization procedure, we consider all of the available methods in our library (Section V), and we capture the inference accuracy on the test dataset by using the quantized model (lines 6-8). If a user-defined accuracy loss threshold is satisfied, then the quantized model is the output of our algorithm (line 9). If a desired accuracy loss threshold is not available, we iterate over all the quantization methods to select the one that delivers the highest accuracy.

Algorithm 1 Aging-Aware Quantization

Input: 1. Synthesized Netlist, 2. Aging Level: e.g., ΔV_{th},
3. Trained Model & Test Dataset, 4. Accuracy Loss Threshold: e
Output: Aging-Aware Quantized Model
1: List = []
2: **for all** (α, β) $\in$ [0, 8]^2:
3: Run STA with (corresponding aging library, compression (α, β))
4: **if** timing constraint is met**:** List $\leftarrow$ add (α, β)
5: (α, β) $\leftarrow$ (α, β) in List with min$\left(\sqrt{\alpha^2+\beta^2}\right)$
6: **for all** method **in** Quantization Library
7: Quantize Model using method and size (8-a, 8-b)
8: Capture accuracy on test dataset
9: **if** threshold e is satisfied**: return** Quantized Model

The (α, β) values that are extracted during the timing analysis phase ensure that the timing constraint is met. Hence, no aging-induced timing errors occur and accurate computations are performed on the compressed inputs. In that way, the inference accuracy is defined *only* by the accuracy delivered by quantization for the respective compression values. *Therefore, the inference accuracy can be captured purely at software level, without the need to perform time-consuming post-synthesis timing simulations that are infeasible for large datasets* [13]. In addition, the padding selection does not affect the quantization process/accuracy, and only affects how the data will be stored in memory (see Section V). Note that the α and β values depend on the NPU microarchitecture (e.g., MAC size) and the aging period. On the other hand, the selected quantization method depends on both the (α, β) and the NN. Hence, for a specific aging period, different NNs will feature the same (α, β) while they might utilize a different quantization method.

Targeting to minimize the employed compression, in line 5 of Algorithm 1, we select the (α, β) that minimizes $\sqrt{\alpha^2+\beta^2}$, i.e., we use the Euclidean distance from (0, 0) as a surrogate model of the applied compression. To evaluate the efficiency of our model in estimating the applied compression we run the following experiment. For each quantization method in our library and for each NN examined in Section VII, we quantize the NN using the respective method and (α, β) compression and then, we capture the accuracy loss w.r.t. the FP32 model. This procedure is repeated $\forall(\alpha, \beta) \in [0,4]^2$. Next, we rank ($\alpha$, β) based on i) the computed accuracy loss, and ii) our model. Finally, we calculate the Pearson Correlation Coefficient between the two rankings obtained. Over the ten examined NNs and the five quantization methods, the Pearson Coefficient is 0.84 on average (ranging from 0.71 to 0.92). Hence, our

TABLE I: Achieved accuracy and selected quantization method for varying NNs at various aging levels (represented by ΔV_{th}).

	Accuracy Loss (%) / Quantization Method Selected				
Neural Network	**10mV**	**20mV**	**30mV**	**40mV**	**50mV**
ResNet50	0.27 / M5*	0.36 / M5	0.97 / M3*	1.47 / M4*	2.37 / M4
ResNet101	0.26 / M5	0.36 / M5	1.28 / M5	0.97 / M4	1.84 / M4
ResNet152	0.28 / M5	0.34 / M5	1.08 / M5	1.12 / M4	2.10 / M4
VGG13	0.15 / M4	0.22 / M3	0.39 / M3	1.20 / M4	2.54 / M4
VGG16	0.05 / M5	0.14 / M5	0.29 / M3	0.73 / M4	1.09 / M4
VGG19	0.20 / M3	0.33 / M3	0.46 / M3	1.09 / M4	2.37 / M4
Alexnet	0.28 / M5	0.54 / M5	0.99 / M5	2.72 / M4	4.00 / M4
SqueezeNet 1.1	0.55 / M5	1.51 / M5	3.61 / M4	6.03 / M4	7.83 / M4
Wide ResNet50	0.14 / M5	0.24 / M5	0.67 / M5	1.27 / M4	2.49 / M4
Wide ResNet101	0.23 / M5	0.41 / M5	1.33 / M5	1.41 / M4	2.92 / M4

* M3: LAPQ [19], M4: ACIQ [18], M5: ACIQ w/o bias correction [18]

TABLE II: The extracted compression values (α,β) and padding for the examined aging levels.

Aging [ΔV_{th}]	**10mV**	**20mV**	**30mV**	**40mV**	**50mV**
(α, β) / Padding	(2,0)/LSB	(2,2)/MSB	(3,1)/LSB	(2,4)/LSB	(3,4)/LSB)

model achieves a very strong correlation in ranking the (α, β) compression values. As an alternative, we can evaluate lines 6-8 for all the extracted (α, β) values. However, considering that some quantization methods are time-consuming, this would heavily impact the execution time of our implementation. On the other hand, by considering only one (α, β) during quantization, only 1 hour was required, in the worst case.

VII. Experimental Results and Evaluation

In order to evaluate the effectiveness of our technique in eliminating the aging-induced timing errors in NPUs, we examine the delay gain delivered by our technique as well as the respective accuracy loss that has to be traded due to the applied input compression. For our evaluation, we consider the Edge TPU microarchitecture [2] and we use the MAC unit described in Section IV (i.e., 8-bit multiplier/22-bit adder) as our driving circuit. In addition, we consider ten NNs (listed in Table I) with varying characteristics. For aging analysis, we use the libraries and workflow described in Section VI-A. All the NNs are trained on the ImageNet dataset [23] and their implementation is based on official PyTorch repositories (Torchvision) [24]. Hereafter, when referring to the baseline design, we refer to the MAC unit when using 8-bit quantization for activations and weights [1] (i.e., no compression α=β=0). Moreover, the accuracy loss is calculated with respect to the accuracy achieved with FP32 inference. Therefore, even the baseline with no-aging will exhibit a small (negligible) accuracy loss. In our algorithm, we do not set an accuracy loss threshold but instead, we iterate over all the quantization methods to select the one that delivers the highest accuracy. For aging, we examine 10 years as the typical projected lifetime. Finally, for our analysis, we also evaluated precision scaling by applying LSB masking on the 8-bit quantized NNs [10], [11]. However, without retraining precision scaling delivered unacceptable accuracy loss for all the examined NNs and aging levels. Considering, that DNN retrain is very time consuming [13], precision scaling [10], [11] is not included in our discussion.

Table II reports the compression values (α,β) and padding as extracted by our algorithm for the examined aging periods (i.e., aging is represented by an increasing ΔV_{th} over time). Over the aging periods, Fig. 4a depicts the delay of the MAC when

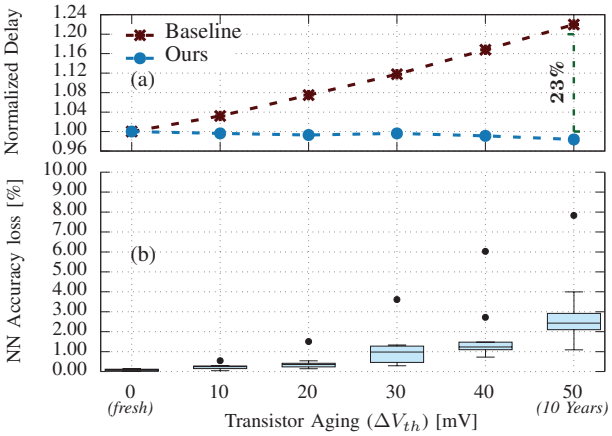

Fig. 4: a) The normalized delay, from the beginning (i.e., fresh) to the end of lifetime (10 years), of the baseline and our approach w.r.t the delay of the fresh baseline. b) Graceful accuracy degradation, over time, delivered by our aging-aware quantization, presented by box plots w.r.t. the examined NNs.

employing our technique (i.e., applying inputs compression as in Table II) in comparison with the delay degradation of the baseline. In Fig. 4a, the delay is normalized with respect to the delay of the fresh baseline (i.e., without aging) without timing guardbands. As shown, the delay of the baseline increases over time (normalized delay is higher than 1), reaching a performance loss of 23% under 10 years aging. This is an expected behavior due to the aging phenomena where transistors become slower. Hence, without timing guardbands, aging-induced timing errors occur which will significantly deteriorate the NPU accuracy (see Fig. 1b). Contrarily, this is not the case when applying our technique. As Fig. 4a demonstrates, our approach does not follow the baseline degradation style. *This is because our technique adaptively compresses the inputs over time to gracefully compensate the delay increase due to aging.* The normalized delay is always less than or equal to 1, making our aging-aware quantization technique resilient against aging-induced delay degradation. Therefore, considering that the achieved delay can be always similar, if not even better, to the delay without aging, our technique effectively suppresses the aging effects and no timing errors will occur. Importantly, our technique shows almost fixed delay from day-zero until the end of the projected lifetime, and hence no guardband is required. Thus, for 10 years lifetime, by removing the timing guardband, a 23% delay gain is achieved compared to baseline.

Nevertheless, our approach results in accuracy degradation since lower range is used for the weights and activations. For the examined NNs, Fig. 4b depicts the accuracy degradation over aging when applying our aging-aware quantization (see Table II). The (α,β) and padding in Table II refer to worst-case delay analysis and hence their selection depends solely on the aging period. Thus, the same compression is used for all the NNs. Note that, since our technique eliminates the timing errors, the accuracy of the aged NPU matches the accuracy achieved by the respective exact quantized model (i.e., (α,β) compression and no-aging). In Fig. 4b, the accuracy loss is presented by box plots over the examined NNs and aging

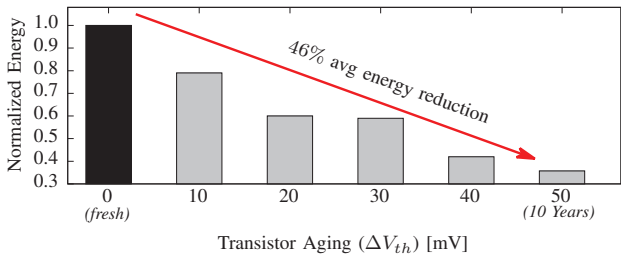

Fig. 5: Normalized energy consumption of our technique over the baseline for varying aging levels. Aging timing guardband is used for the baseline to prevent timing errors.

periods. As shown, our technique delivers graceful accuracy degradation over time. For example, the average accuracy loss is 0.24%, 0.45%, 1.11%, 1.80%, and 2.96% for aging (ΔV_{th}) 10mV, 20mV, 30mV, 40mV, and 50mV, respectively, where 50mV is equal to 10 years aging. In addition, as shown in Fig. 4b, for all the examined periods, the accuracy loss is well concentrated around the median demonstrating that our technique is slightly affected by the NN model. The highest reported accuracy loss is 7.83% for 10 years aging for the SqueezeNet network. SqueezeNet features always the highest accuracy loss for all aging periods. SqueezeNet is by design a very compressed network and thus, it is affected considerably by low bit-width quantization. The above analysis examines timing guardband elimination. However, with (3,1) compression and only 9% guardband the accuracy loss becomes 1.11%, on average, for 10 years aging. The full accuracy results are summarized in Table I. In addition, in Table I, we report the quantization method selected by our algorithm for each case. Table I reports the accuracy as obtained from PyTorch for the respective compression and quantization method. As shown in Table I, the quantization method LAPQ [19] is selected in the 14% of the cases, while ACIQ [18] and ACIQ w/o bias [18] are selected in the 44% and 42% of the cases, respectively. The methods [16], [17] were not selected in any aging level since the required compression values (Table II) were very high and out of the effective range of [16], [17]. Table I highlights the importance of considering a quantization methods library since the best quantization method varies with respect to the required compression and the NN model itself.

Finally, we evaluate the energy efficiency of our technique. Over time, our technique adaptively compresses the MAC inputs to eliminate the aging effects. However, input compression leads also to reduced switching activity and thus, lower power consumption. Fig. 5 depicts the normalized energy of our technique w.r.t the energy consumption of the baseline for varying aging levels. In Fig. 5, the MAC unit using our approach is operated at the maximum frequency of the fresh MAC, while for the baseline a 23% timing guardband (Fig. 4a) is used to prevent timing errors due to aging. As shown in Fig. 5, for no aging, our technique does not induce any overhead, while for 10mV-50mV aging, it delivers 46% energy reduction on average (ranging from 21% up to 67%).

VIII. Conclusion

In this paper, we employ quantization, for the first time, to suppress aging in NPUs. Our technique applies adaptive aging-aware quantization over time and achieves a progressive, though small, accuracy degradation, while in the meantime it eliminates the deleterious aging-induced timing errors. Our technique enables removing the aging timing guardbands and boosts the NPU performance, while in the meantime it significantly decreases the energy consumption. Moreover, our approach modifies only the inputs of the NPU and it does not require any modifications to the underlying microarchitecture.

Acknowledgement

Authors would like to thank Yogesh Chauhan and Souvik Mahapatra and their teams for the valuable help in compact model calibration. This work is supported in part by the German Research Foundation (DFG) through the project "ACCROSS: Approximate Computing aCROss the System Stack".

References

[1] N. P. Jouppi, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Int. Symp. on Computer Architecture*, 2017.
[2] S. Cass, "Taking ai to the edge: Google's tpu now comes in a maker-friendly package," *IEEE Spectrum*, vol. 56, no. 5, 2019.
[3] J. Song *et al.*, "7.1 an 11.5 tops/w 1024-mac butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile soc," in *IEEE International Solid-State Circuits Conference*, 2019.
[4] H. Amrouch, G. Zervakis *et al.*, "NPU Thermal Management," in *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2020.
[5] W. Sootkaneung, S. Howimanporn *et al.*, "Thermal effect on performance, power, and bti aging in finfet-based designs," in *DATE*, 2017.
[6] S. Mahapatra, N. Goel *et al.*, "A Comparative Study of Different Physics-Based NBTI Models," *TED*, vol. 60, no. 3, March 2013.
[7] C. Hu, *Modern semiconductor devices for integrated circuits*. Prentice Hall, 2010.
[8] S. Roy, D. Liu *et al.*, "Osfa: A new paradigm of aging aware gate-sizing for power/performance optimizations under multiple operating conditions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 10, 2016.
[9] J. Keane and C. H. Kim, "Transistor aging," in *IEEE Spectr*, vol. 48, April 2011, p. 28–33.
[10] H. Kim, J. Kim *et al.*, "Aging compensation with dynamic computation approximation," *IEEE Trans. Circuits Syst. I*, vol. 67, no. 4, 2020.
[11] H. Amrouch, B. Khaleghi *et al.*, "Towards aging-induced approximations," in *Design Automation Conference*, 2017.
[12] G. Zervakis, H. Amrouch *et al.*, "Design automation of approximate circuits with runtime reconfigurable accuracy," *IEEE Access*, vol. 8, 2020.
[13] Z.-G. Tasoulas, G. Zervakis *et al.*, "Weight-Oriented Approximation for Energy-Efficient Neural Network Inference Accelerators," *TCASI*, 2020.
[14] V. Mrazek, Z. Vasicek *et al.*, "Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining," in *International Conference on Computer-Aided Design (ICCAD)*, Nov 2019.
[15] A. Thirunavukkarasu, H. Amrouch *et al.*, "Device to circuit framework for activity-dependent nbti aging in digital circuits," *IEEE Trans. Electron Devices*, vol. 66, no. 1, 2019.
[16] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.
[17] B. Jacob, S. Kligys *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *CVPR*, 2018.
[18] R. Banner, Y. Nahshan *et al.*, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *NeurIPS*, 2019.
[19] Y. Nahshan, B. Chmiel *et al.*, "Loss aware post-training quantization," *arXiv preprint arXiv:1911.07190*, 2019.
[20] N. Parihar, N. Goel *et al.*, "Bti analysis tool—modeling of nbti dc, ac stress and recovery time kinetics, nitrogen impact, and eol estimation," *TED*, vol. 65, no. 2, 2018.
[21] S. Natarajan *et al.*, "A 14nm logic technology featuring 2nd-generation finfet, air-gapped interconnects, self-aligned double patterning and a 0.0588 µm2 sram cell size," in *Int. Electron Devices Meeting*, 2014.
[22] S. Mishra, H. Amrouch *et al.*, "A simulation study of nbti impact on 14-nm node finfet technology for logic applications: Device degradation to circuit-level interaction," *TED*, vol. 66, no. 1, 2018.
[23] J. Deng, W. Dong *et al.*, "Imagenet: A large-scale hierarchical image database," in *Conf. on computer vision and pattern recognition*, 2009.
[24] Pytorch, "Torchvision models, https://pytorch.org/docs/stable/torchvision/models.html."