Built-in Self-Test and Built-in Self-Repair Strategies Without Golden Signature for Computing in Memory

Yu-Chih Tsai, Wen-Chien Ting, Chia-Chun Wang, Chia-Cheng Chang, Ren-Shuo Liu
National Tsing Hua University, Hsinchu, Taiwan
{williemiku0504, kevinting0325, k4e520, raymond11022}@gmail.com, renshuo@ee.nthu.edu.tw

***Abstract*—This paper proposes built-in self-test (BIST) and built-in self-repair (BISR) strategies for computing in memory (CIM), including a novel test method and two repair schemes. They all focus on mitigating the impacts of inherent and inevitable CIM inaccuracy on convolution neural networks (CNNs). Regarding the proposed BIST strategy, it exploits the distributive law to achieve at-speed CIM tests without storing testing vectors or golden results. Besides, it can assess the severity of the inherent inaccuracies among CIM bitlines instead of only offering a pass/fail outcome. In addition to BIST, we propose two BISR strategies. First, we propose to slightly offset the dynamic range of CIM outputs toward the negative side to create a margin for negative noises. By not cutting CIM outputs off at zero, negative noises are preserved to cancel out positive noises statistically, and accuracy impacts are mitigated. Second, we propose to remap the bitlines of CIM according to our BIST outcomes. Briefly speaking, we propose to map the least noisy bitlines to be the MSBs. This remapping can be done in the digital domain without touching the CIM internals. Experiments show that our proposed BIST and BISR strategies can restore CIM to less than 1% Top-1 accuracy loss with slight hardware overhead.**

***Index Terms*—computing in memory, built-in self-test, built-in self-repair, accelerator, convolutional neural networks**

I. Introduction

Recently, computing in memory (CIM) [1]–[3] has been an emerging architecture for convolutional neural networks (CNNs). As an alternative to Von-Neumann architectures, CIM provides the potentials of data movement reduction, low power consumption, and better computing throughput. Most CIM augments conventional memory with multiply–accumulate (MAC) computing capability to support CNNs. That is, bitline $Out_k = \sum_i A_i \times W_{i,k}$ (where $W_{i,k}$ means the k-th bit of W_i), and $P_{sum} = \sum_k (Out_k \times 2^k)$.

Having said that, typically, CIM inevitably exhibits inaccuracies due to its analog computing nature. Thus, the actual MAC outputs can be probabilistically deviated from the ideal MAC results. The potential sources of the inaccuracies are enormous and out of the scope of this paper. For example, they can come from the process variations (e.g., transistor size variations among memory cells, parasitic capacitance variations among bitlines, reference source variations among analog-to-digital converters (ADC)) and randomness (e.g., thermal noises and random telegraph noises).

Considering CIM's inherent inaccuracies, we believe that built-in self-test (BIST) and built-in self-repair (BISR) strategies are the musts for CIM to mitigate the impacts of inaccuracies on CNNs. Although BIST and BISR ideas are common in conventional memory [4], few works have designed BIST and

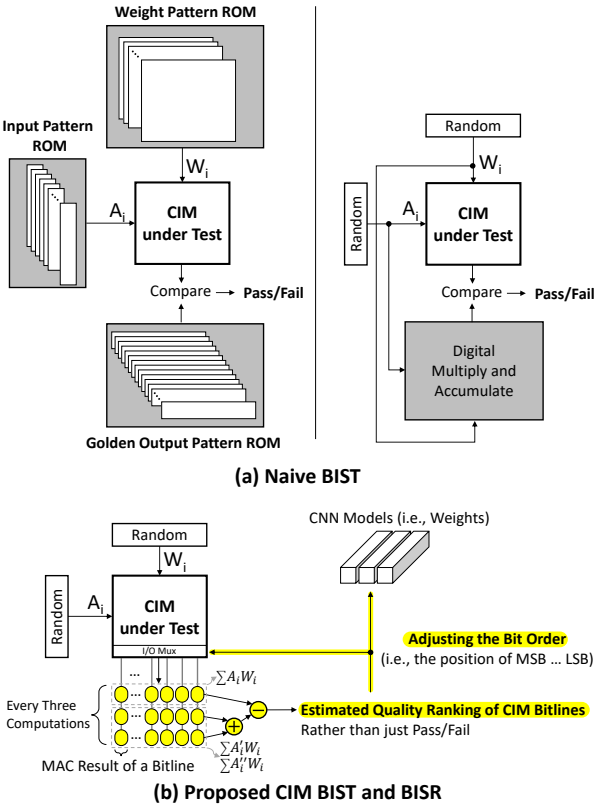

Fig. 1: Proposed BIST and BISR

BISR for CIM, which combines both memory and computing functionalities. As for the conventional memory functionality of CIM, conventional BIST and BISR may be applicable, e.g., to detect and repair stuck-at cells, opens, and shorts. This work does not focus on changing them. As for the MAC computing functionality of CIM, however, conventional BIST and BISR are inapplicable or not very efficient.

Figure 1 shows the high overheads of naively employing the conventional BIST ideas to test CIM. In comparison, our proposed BIST and BISR lead to relatively low overheads. Figure 1(a) is the first naive approach to assess the inaccuracies in CIM. It resembles the conventional Automatic Test Equipment (ATE) method that stores testing sequences of A_i and W_i, stores their corresponding golden results (also known as the golden signature), and compares the golden outputs with the actual CIM outputs accordingly. Clearly, this approach demands substantial storage. For example, assume each MAC

978-3-9819263-7-8/DATE23/© 2023 EDAA

operation on one CIM bitline involves 16 4-bit activations. The total number of different input vectors is $2^{(4\times16)} = 2^{64}$. This number suggests a large number of input vectors and corresponding golden results are needed to achieve sufficient coverage, and they cause significant on-chip storage overhead. Although using a multiple input shift register (MISR) [5] can compress the golden results, it hinders the capability of assessing the severity of CIM's inaccuracy. One may think of storing the test patterns off-chip, e.g., in the ATE machine, but doing so hinders at-speed tests and the ATE machines lead to a production bottleneck.

Figure 1(a) also shows another naive way to assess the inaccuracies in CIM by randomizing A_i and W_i and generating the golden results on-the-fly using a golden MAC circuit, typically a digital MAC circuit. The overheads of this approach come from the golden digital MAC circuit. If one wants to perform at-speed testing on CIM, the speed and parallelism of the golden digital MAC circuit need to be comparable to the CIM under test. According to the rationale for adopting CIM in accelerators, CIM should achieve much higher area efficiency and power efficiency than the digital counterpart. Therefore, employing an extra golden digital MAC circuit for CIM BIST is unreasonable.

Algorithm-based fault tolerance (ABFT) for matrix operations [6] is another method to detect and correct errors. ABFT does not directly support assessing the severity of CIM's inaccuracy. In addition, ABFT has limited error correction capacity and does not support graceful degradation, which our design supports.

In this paper, we firstly propose a novel BIST method for measuring the inaccuracies in CIM by utilizing the distributive law. We exploit an invariant that $\sum_i A_i \times W_{i,k} = \sum_i A'_i \times W_{i,k} + \sum_i A''_i \times W_{i,k}$ as long as $A_i = A'_i + A''_i$. We propose to randomize A_i and W_i, randomly split A_i into A'_i and A''_i, and get the three CIM outputs associated with A_i, A'_i, and A''_i, accordingly, as illustrated in Figure 1(b). Doing so allows us to check the distributive law among the three outputs to estimate the CIM computation quality instead of only getting a pass/fail result. Furthermore, we can also rank the estimated computation qualities of the CIM bitlines. The proposed method follows a straightforward concept and does not incur excessive overheads. First, it avoids the storage overhead of storing test vectors because randomizing A_i and W_i can be achieved on-the-fly. Randomizing A_i and W_i can be done using linear feedback shift register (LFSR) [7], and randomly splitting A_i can be done using LFSR and AND circuits. Second, it does not require another golden MAC circuit or store the golden result. These benefits make our approach a suitable BIST design for CIM. BIST is a must for CIM, and compared with the naive approaches in Figure 1(a), our proposal incurs reasonable overheads.

In addition to measuring the CIM inaccuracies, we explore the impacts of such CIM inaccuracies on the accuracy of AI tasks, e.g., image classification tasks using CNNs. Experiments show sharp degradation in the accuracy of AI tasks once CIM exhibits a certain level of inaccuracy. In response, we propose restoring the accuracy of AI tasks by combining BIST with BISR.

We propose two BISR strategies to counter the inaccuracy. First, we observe that CIM's inaccuracy causes the distributions of CNNs' output activations to distort, and we find that this problem can be dealt with by offsetting the dynamic range of CIM outputs. For example, let us consider a CIM bitline performing the MAC operation which includes an N-bit output ADC. Thus, its output range is $[0, 2^N - 1]$. Given the same dynamic range, we propose to offset the output range from $[0, 2^N - 1]$ to $[-1, 2^N - 2]$. It helps to preserve negative noises that can cancel out positive noises since AI tasks accumulate many such MACs. Otherwise, cutting off such negative noises makes the noise distribution at zero become asymmetrical.

Second, as shown in Figure 1(b), we propose a BISR strategy that utilizes the results obtained from our BIST by reordering CIM bitlines and reordering the weight bits of a CNN model correspondingly. More specifically, it allocates CIM bitlines exhibiting the least inaccuracies to handle higher-weighting bits, i.e., the most-significant bits (MSBs) in CNNs. The overheads of this strategy are reasonable because reordering CIM bitlines is done in the digital domain by multiplexers, and reordering the weight bits of a CNN model is a simple number conversion pass without involving training, re-training, or any training procedure.

We perform image classification tasks using Resnet-50 [8] and VGG-16 [9] on the ImageNet [10] dataset with our CIM simulator to test our proposed BIST and BISR. Our BIST method can successfully assess the severity of CIM output inaccuracies, and our BISR strategies also show significant accuracy restoration of less than 1% Top-1 accuracy loss compared with ideal CIM. We also implement the proposed BIST and BISR and compare with CIM-based accelerator; the result verifies that the proposed BIST and BISR are hardware friendly with only slight hardware overhead.

The rest of this paper is organized as follows: Section II shows the background and related works. Section III details our proposed system-level design, BIST and BISR strategies. Finally, Section IV evaluates our proposed strategies and presents the experiment results.

II. Background

A. MAC Operations in Computing in Memory

This paper focuses on CIMs that perform multiply and accumulate operations, widely used in AI operations such as convolutions. Figure 2 shows how a typical CIM performs the MAC operation; multiple activations are multiplied by weight bits stored in different bitlines separately in the CIM. In this example, sixteen 4-bit activations are multiplied by sixteen 8-bit weights. The CIM performs the MAC operations of activations and weights' each bit individually, i.e., sixteen 4-bit activations are multiplied with sixteen weights' single bit and summed along each CIM bitline. It also shows the noise in each bitline, which will be discussed in the following subsection. Since we assume an adequate number of bits is allocated for each bitline, each bitline is represented as an 8-bit unsigned integer. Outputs

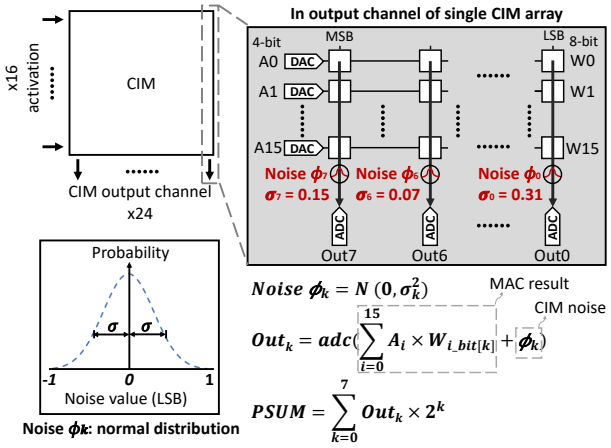

Fig. 2: MAC operation and simulated noise injection in CIM.

of each bitline are then weighted with powers of 2, i.e., left-shifted by their corresponding number of bits.

This paper adds a positive offset (per channel of CNNs) to all weights in convolutional layers to make weights positive or zero, allowing convolution results to be acquired using unsigned MAC operations. This offset, known as the "zero-point," is deducted from the partial sum of each MAC operation independently.

B. CIM Noise Model

Various sources of inaccuracies in CIM architectures are discussed in [11]. As stated in [12] and [13], ADCs typically have thermal noises with Gaussian distribution in the time domain, along with various independent noise sources in the CIM cells and noises in the charge summing or current summing process. Therefore, this paper focused on CIM inaccuracies modeled as additive Gaussian noises in the analog domain.

As shown in Figure 2, we inject continuous Gaussian noises σ before round and clip operations to simulate the effect of ADC outputs. Our simulated CIM has 24 parallel MAC arrays (192 CIM bitlines), and each CIM bitline is associated with a fixed standard deviation. The standard deviation values were randomly selected and uniformly distributed between 0 and $0.35\ LSB$. As [14] mentioned that the average standard deviation is $0.37\ LSB$ and hence we choose $0.35\ LSB$ as the upper bound. In addition, experiments also simulated with higher Gaussian standard deviations upper bounds $0.45\ LSB$ and $0.55\ LSB$.

C. CNN Robustness

A common way to enhance the robustness of CNN is by retraining the model; this commonly involves incorporating noises or errors previously nonexistent into the training set, such as in [15], [16]. However, noises and errors in CIM may vary from chip to chip, making it difficult to predetermine the inaccuracies and incorporate them into the training process. Therefore, some work develops an in-memory calibration scheme for noises due to chip variation [17].

Our testing method can be a means of retrieving such errors needed for retraining the model. In addition, our BISR methods are alternatives to recovering the lost accuracy without retraining the model.

III. Proposed BIST and BISR Strategies

A. System-Level Block Diagram

Figure 3 shows a system-level block diagram of our proposed BIST and BISR. The block diagram consists of three primary parts: **1)** an on-chip global SRAM buffer where weights and activations are stored, **2)** the CIM, which in this case is our device under test, and **3)** our proposed BIST and BISR architecture. The BIST and BISR procedure performs once when the chip is manufactured, and the inaccuracy order of each CIM bitline is recorded in the reordering table. This information is used to perform weight bit reordering and CIM bitline reordering in the BISR process.

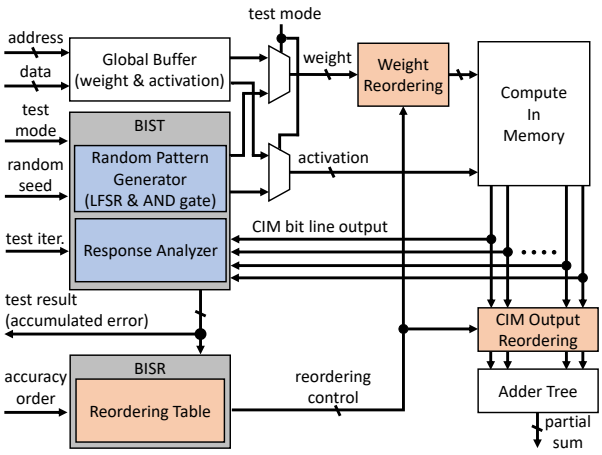

Fig. 3: System-level architecture

B. Built-in Self-Test Strategy

To tackle the storage overhead of test vectors, the hardware overhead of golden MAC circuits, and estimating the CIM computation quality, we developed a test procedure as Algorithm 1 that utilizes the distributive property. We exploit an invariant that $\sum_i A_i \times W_{i,k} = \sum_i A'_i \times W_{i,k} + \sum_i A''_i \times W_{i,k}$ as long as $A_i = A'_i + A''_i$. That means the bitline has an error when $result_A \neq result_{A'} + result_{A''}$. In the end, the accumulated total errors can be used to profile each CIM bitline as inaccuracy indices. Random weights, activations, and bitmasks are used to test the CIM under conditions similar to perform MAC operations to accelerate AI tasks since our CIM is not dedicated to only a particular model or CNN layer.

Equations (1)-(5) explain how it works more formally. If we ignore the round and clip effect of the ADCs, ϕ_{noise} is a random variable of zero mean normal distributions with standard deviation σ:

$$result_x = MAC_x + \phi_{noise}\,;\quad \phi_{noise} \sim \mathcal{N}(0, \sigma^2) \tag{1}$$

$$result_A - (result_{A'} + result_{A''}) = \phi_{noise} - (\phi_{noise} + \phi_{noise}) \tag{2}$$

Algorithm 1 BIST Procedure

Input: random seed, N (number of test iterations)
Output: accumulated error

for $iteration = 1, 2, ...N$ **do**
  Initialize weights $W_0, \dots W_{15} \leftarrow random$
  Set activations $A_0 \dots A_{15} \leftarrow random$
  Set bitmask $M_0 \dots M_{15} \leftarrow random$
  Use CIM to compute $\Sigma A_i \times W_{i,bit\_k}$ and get $result_A$ (1_{st} MAC operation)
  Set activations $A_0^{'} \dots A_{15}^{'} \leftarrow A_i \& M_i$
  Use CIM to compute $\Sigma A_i^{'} \times W_{i,bit\_k}$ and get $result_{A'}$ (2_{nd} MAC operation)
  Set activations $A_0^{''} \dots A_{15}^{''} \leftarrow A_i \& \overline{M_i}$
  Use CIM to compute $\Sigma A_i^{''} \times W_{i,bit\_k}$ and get $result_{A''}$ (3_{rd} MAC operation)
  $error \leftarrow error + |result_A - (result_{A'} + result_{A''})|$
end for
$accumulated\ error \leftarrow error$
return accumulated error

A linear combination of $\phi_{noise} - (\phi_{noise} + \phi_{noise})$ is still a random variable of a zero-mean normal distribution:

$$\phi_{noise} - (\phi_{noise} + \phi_{noise}) \sim \mathcal{N}(0, 3\sigma^2) \tag{3}$$

The error term is the absolute value of this random variable, making it a random variable of a folded normal distribution:

$$error = |\phi_{noise} - (\phi_{noise} + \phi_{noise})| \sim \mathcal{FN}(0, 3\sigma^2) = Y \tag{4}$$

Since $\mu = 0$, the mean of this folded normal distribution is proportional to σ:

$$\mu_Y = \sigma\sqrt{\frac{6}{\pi}}e^{(-\mu^2/6\sigma^2)} + \mu(1 - 2\Phi(-\frac{\mu}{\sqrt{3}\sigma})) \propto \sigma(\mu = 0) \tag{5}$$

The accumulation of this error term should reflect the magnitude of σ, which represents the severity of the Gaussian noise. Even with ADC's round and clip effect, a correlation between BIST results and the standard deviation of Gaussian noise is still establishable.

Compared with two naive BISTs in Figure 1(a), our BIST strategy uses a random pattern generator instead of massive storage. Additionally, our BIST strategy only needs adders and accumulators as the response analyzer instead of expensive golden digital MAC circuits. These two features make our BIST more hardware friendly.

C. Built-in Self-Repair Strategy

Slight noises ($\sigma \leq 0.35\ LSB$) in CIM cause the Top-1 accuracy of ResNet-50 drops from 77.7% to 17.7%. We speculate that two primary reasons cause such drastic accuracy degradation: **1)** the noises in CIM distort the distribution of output activations, and **2)** the effects of noises are amplified by the shift operations applied to each CIM output.

To tackle these two problems above, we propose two corresponding strategies to reduce the impact of such inaccuracies: **1) CIM Output Range Adjustment**. **2) CIM Weight Bit and Bitline Reordering** such that CIM bitlines with higher inaccuracies can be used to compute less significant weight bits of the MAC operation.

D. CIM Output Range Adjustment

Figure 4 shows the distribution of output activations in the 45_{th} layer of the ResNet-50 model with ideal and noisy CIM. Two noisy CIM have executed ADC clip operation (output range $[0, 255]$ and output range $[-1, 254]$), and another noisy CIM has removed the clip operation (unlimited output range). It is evident that the distribution with ADC clip operation and normal output range $[0, 255]$ has severely distorted; this phenomenon tends to become more significant when using CIM with inevitable noises to perform AI tasks.

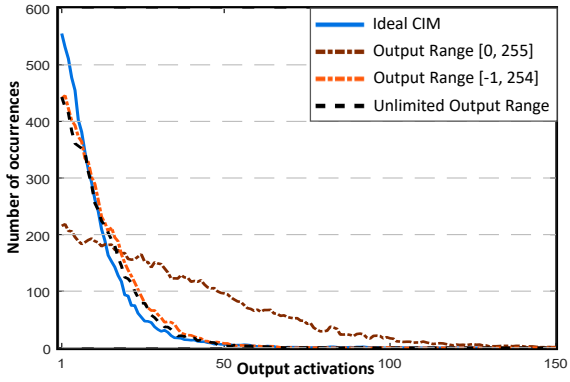

Fig. 4: Distribution of output activations in the 45_{th} layer of ResNet-50 when using ideal and noisy CIMs

A straightforward explanation is that when MAC results before ADC are close to zero, only positive noises affect the CIM output because ADCs clip negative noises to zero. That means Gaussian noises are not symmetrical anymore, and the positive noises are accumulated during activations propagate to the following layers, and eventually cause accuracy loss. Although the distortion can be avoided by removing the clip operation, in reality, it cannot be removed since a finite number of bits are allocated for each CIM output. However, the distortion can be avoided by loosening the restriction of having only CIM output values greater than or equal to zero, i.e., MAC results below the lower bound of ideal CIM can be uniquely represented for keeping the symmetry of Gaussian noises. This paper proposes an alternative method to adjust the equivalent CIM output range to $[-1, 254]$.

The goal mentioned above can be achieved by adjusting the voltage reference of ADCs in CIM by a negative offset of one step size. As a result, the new CIM output code with a particular reference voltage is greater than its original CIM output code by one step size, making the equivalent CIM output range $[-1, 254]$. So, for example, if the new CIM output code is 0, its original CIM output code is -1. We can then deduct the sum of this offset from the partial sum afterward so that an unsigned representation of the MAC results is maintained. That means it allows CIMs to present little negative values, thus making the

noises maintain symmetry when the MAC results are close to zero. Figure 4 shows the similarity in the distribution of the adjusting CIM output range $[-1, 254]$ and the unlimited output range, which validates the effectiveness of our compromised method.

E. CIM Weight Bit and Bitline Reordering

The second source of accuracy drop is the elevated noises effect when the noisy MAC results are shifted. Noises in the MSB CIM bitline degrade the classification accuracy much more severely than noises in the LSB CIM bitline because noises are left-shifted more bits.

To reduce the impact of this problem, we propose the following BISR strategy using this property, as shown in Figure 5. **1)** Accumulated errors from our proposed BIST procedure can be retrieved and used as indicators of inaccuracy severity to determine the inaccuracy order of each CIM bitline. **2)** Weights are reordered on a bitwise level according to the inaccuracy order determined in the previous step. **3)** CIM bitlines are mapped to their corresponding number of bits to be left-shifted.

We defined the reordering levels: Level-1 selects the least inaccurate CIM bitline and assigns it to compute the most significant bit of the MAC operation. Level-2 selects the top two least inaccurate CIM bitlines and assigns them to compute the two most significant bits of the MAC operation, and so on.

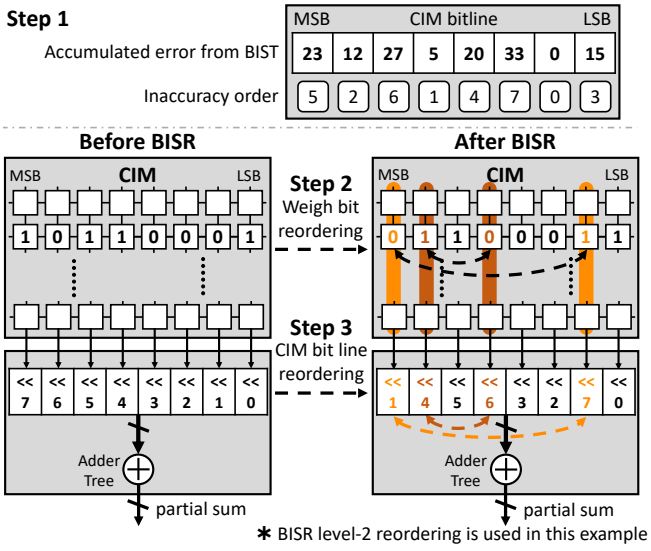

Fig. 5: BISR with CIM weight and bitline reordering

IV. Experimental Results

Experiment results verified our proposed BIST and BISR strategies by performing image classification tasks using 8-bit activations and 8-bit weights models, including Resnet-50 and VGG-16 on the ImageNet dataset with our CIM simulator. Our CIM simulator performs the operational behavior of CIM and injects the Gaussian noises with user-defined σ in run-time. We also implement our proposed BIST and BISR to evaluate the hardware overhead.

A. BIST Result

This experiment aims to establish a correlation between the accumulated errors acquired using our BIST testing method and the severity of Gaussian noises. Our test iteration is 100, and the number of experiment statistics is 1000.

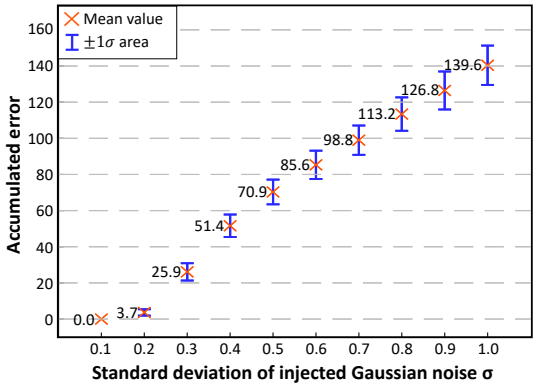

Fig. 6: Correlation between accumulated error and standard deviation of injected Gaussian noise

As shown in Figure 6, it is evident that there is a positive correlation between the total error acquired in the BIST procedure and the severity of the Gaussian noise.

The main difference between test iterations is how much the error standard deviations can be separated. Therefore, a higher test iteration allows us to get a relatively more accurate estimate of the noise severity and require a longer testing time. In later experiments, our test iteration is 100 because the Gaussian noises σ are smaller than $0.55\,LSB$ which is clear to separate.

B. CIM Output Range Adjustment and Reordering Result

To evaluate the effectiveness of CIM output range adjustment and reordering repair strategies on accuracy restoration, we simulated using noisy CIM to accelerate image classification tasks which are ResNet-50 and 1000 test samples.

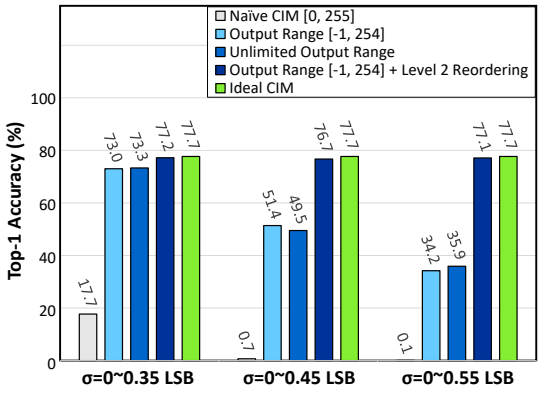

Fig. 7: ResNet-50 accuracy with proposed BISR

First, Figure 7 shows that accuracy can immediately be restored by adjusting the CIM output range and that CIM output range $[-1, 254]$ and unlimited output range have similar effects on accuracy restoration. However, there is still a considerable

gap in accuracy between unlimited CIM output range and ideal CIM, indicating the limitations of this strategy. Second, Figure 7 shows that the Top-1 accuracy can be significantly restored with less than 1% accuracy loss when CIM output range $[-1, 254]$ and level-2 reordering are both applied, which are our proposed repair strategies.

C. BIST and BISR with More Models Result

This experiment aims to evaluate proposed BIST and BISR strategies with more noise parameters, CIM weight bit and bitline reordering, and models including ResNet-50, VGG-16, and 50000 test samples are shown in this section.

We approximated the effects of CIM output range adjustment with an unlimited CIM output range to speed up our experiments. We have two reasons show this approximation is reasonable. First, the distribution of output activations when the CIM output range is set to $[-1, 254]$ is similar to that of the unlimited output range, as shown in Figure 4 of Section III-D. Second, experiment results shown in Section IV-B confirm this assumption as accuracy restoration is still present without an unlimited output range.

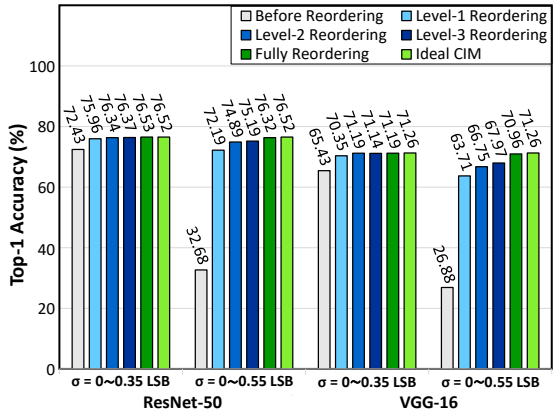

Fig. 8: More models' accuracy with proposed BIST and BISR

Figure 8 shows that noisy CIM with proposed BIST and BISR strategies can be significantly restored with less than 1% Top-1 accuracy loss with fully (level-8) reordering. When the CIM with slight Gaussian noises ($\sigma \leq 0.35 LSB$), the accuracy drop can be within 1% even with level-1 reordering.

D. Implementation Result

TABLE I: Area percentage of CIM-based accelerator

Category	Area Percentage
CIM Macro	18.92 %
Data Buffer (SRAM)	30.13 %
Digital Circuit without BIST and BISR	47.84 %
Random Pattern Generator of Proposed BIST	**0.51 %**
Response Analyzer of Proposed BIST	**0.41 %**
Proposed BISR	**2.18 %**

We implement the proposed BIST and BISR capable of fully (level-8) reordering CIM weight bits and bitlines, and compare with CIM-based accelerator. The area percentage of each circuit is shown in Table I. It is noteworthy that the proposed BIST is hardware friendly with less than 1% area overhead because our response analyzer only needs adders and accumulators instead of golden digital MAC circuits. Our random pattern generator also only needs LFSR and AND circuits. The proposed BISR can restore CNNs accuracy with a slight area overhead of approximately 2%.

V. Conclusion

In this paper, novel BIST and BISR strategies for CIM are proposed. The proposed self-testing method is more hardware friendly and can successfully profile CIM outputs with accumulated errors, which can be used as a quality metric for CIMs. The first proposed self-repairing strategy adjusts the range of CIM outputs, aiming to avoid distortion of output activations, which would cause massive accuracy loss. The second proposed self-repairing strategy reorders CIM bitlines and corresponding weight bits to avoid noisier CIM bitlines being shifted by a more significant number of bits.

Experiment results validate how CIM inaccuracies would impact image classification tasks and demonstrate the effectiveness of our proposed BIST and BISR strategies, in which Top-1 accuracy loss is less than 1%.

Acknowledgments

We thank reviewers for their valuable comments. We also thank National Center for High-performance Computing (NCHC) and Taiwan Semiconductor Research Institute (TSRI) for providing computational and storage resources. This work is supported in part by National Science and Technology Council (NSTC) projects 111-2823-8-007-001, 111-2218-E-007-009, and 110-2218-E-007-037 and TSMC Ph.D. scholarship.

References

[1] J. -W. Su et al., "16.3 A 28nm 384kb 6T-SRAM computation-in-memory macro with 8b precision for AI edge chips," ISSCC'21.
[2] J. Wang et al., "14.2 A compute SRAM with bit-serial integer/floating-point operations for programmable in-memory vector acceleration," ISSCC'19.
[3] Q. Liu et al., "33.2 A fully integrated analog ReRAM based 78.4TOPS/W compute-in-memory chip with fully parallel MAC computing," ISSCC'20.
[4] Jin-Fu Li et al., "A built-in self-repair design for RAMs with 2-D redundancy," IEEE Transactions on Very Large Scale Integration Systems, vol. 13, no. 6, pp. 742-745, June 2005.
[5] D. K. Pradhann et al., "A new framework for designing and analyzing BIST techniques and zero aliasing compression," IEEE Transactions on Computers, vol. 40, no. 6, pp. 743-763, June 1991.
[6] Kuang-Hua Huang et al., "Algorithm-based fault tolerance for matrix operations," IEEE Transactions on Computers, vol. C-33, no. 6, pp. 518-528, June 1984.
[7] M. Mohan et al., "Review on LFSR for low power BIST," ICCMC'19.
[8] Kaiming He et al., "Deep residual learning for image recognition," arXiv:1512.03385, 2015.
[9] Karen Simonyan et al., "Very deep convolutional networks for large-scale image recognition," in arXiv:1409.1556, 2015.
[10] Jia Deng et al., "ImageNet: a large-scale hierarchical image database," CVPR'09.
[11] S. K. Gonugondla et al., "Fundamental limits on the precision of in-memory architectures," ICCAD'20.
[12] Bryan Lizon, "Fundamentals of precision ADC noise analysis," in Texas Instruments website, Sep. 2020.
[13] A. S. Rekhi et al., "Analog/mixed-signal hardware error modeling for deep learning inference," DAC'19.
[14] BH. Jia et al., "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," JSSC, vol. 55, no. 9, pp. 2609-2621, Sep. 2020.
[15] K. Matsuoka, "Noise injection into inputs in back-propagation learning," IEEE Transactions on Systems, Man, and Cybernetics, vol. 22, no. 3, pp. 436-440, May-June 1992.
[16] S. K. Gonugondla et al., "A variation-tolerant in-memory machine learning classifier via on-chip training," JSSC, vol. 53, no. 11, pp. 3163-3173, Nov. 2018.
[17] J. Kim et al., "Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array," Symposium on VLSI Circuits, 2019.