GEMMHeal: An Efficient Self-repairing Architecture for Matrix Multiplication

Yuxuan Qiao[†,‡,*], Changxu Liu[†,‡,*], Hao Zhou[†,‡], Yifei Feng[§], Yifan Song[†,‡], Junjie Zuo[†,‡] and Fan Yang[†,‡]
[†] *State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai, China*
[‡] *School of Microelectronics, Fudan University, Shanghai, China*
[§] *School of Computer Science, Fudan University, Shanghai, China*

***Abstract*—In modern machine learning models like Transformers, matrix multiplication dominates most computation. Specific hardware often uses large-scale PE arrays, such as systolic arrays, to accelerate this process. However, these extensive PE arrays tend to experience high fault rates, with cumulative errors in matrix multiplication potentially impacting final algorithmic outcomes. We propose GEMMHeal, a self-repairing, systolic-array-like architecture optimized for matrix multiplication. This architecture optimizes the matrix multiplication array design and data flow in the buffer while decoupling the error correction (EC) core from the array. The EC core operates as an independent computation unit, maintaining resource efficiency and avoiding performance degradation due to the spatial locality of faults. Additionally, our proposed design incorporates redundancy within the EC core, allowing it to self-repair and thus further improve fault tolerance. GEMMHeal is implemented in a 32nm process to demonstrate its area and physical achievability. A single EC core occupies only 1.8% of the area while effectively repairing up to 6.3% of faulty PEs, all while maintaining overall performance. By accounting for potential faults within the EC core, our redundant design achieves an average improvement of 31% in the EC core normal rate compared to non-redundant designs when the computation cell fault rate is within 15%.**

***Index Terms*—Matrix Multiplication, AI Accelerator, Systolic Array, Redundancy, Self-Repairing**

I. INTRODUCTION

In machine learning applications such as the large language model (LLM), matrix multiplication is crucial and accounts for most of the computation [1]–[3]. Many hardware-based solutions have been developed to accelerate matrix multiplication [4], [5]. These solutions often feature highly integrated structures, which also implies a higher rate of faults, including but not limited to manufacturing defects and run-time failures [6], [7]. Any failure can spread and affect the final results. Although neural networks inherently tolerate computational errors or occasional inaccuracies [8], in certain applications, such as self-driving, hardware errors can lead to incorrect results in algorithmic models, potentially causing security issues [9]. Similarly, large-scale AI computations often involve thousands of accelerators, where failures in critical nodes can disrupt the entire model's availability, potentially causing economic losses in the millions of dollars [10].

Numerous approaches have been proposed to mitigate or prevent the impact of computational faults on matrix multiplication accelerators [11]–[16]. These solutions can tolerate certain levels of hardware failure without affecting the accuracy of computational results. However, these solutions still have some limitations:

- **Increased scheduling complexity or waste existing resources:** Repairing the hardware faults by bypassing faulty PEs or disabling columns containing faulty PEs can significantly increase the complexity of scheduling or waste the computational resources of PEs that have not failed [11], [12].
- **Additional preprocessing overhead or lack generality:** By pruning and sparsifying the matrix, assigning weights that are zero to faulty PEs and skipping the computation of these PEs can tolerate hardware faults without adding extra scheduling overhead [13]. However, this method increases the software processing overhead each time it is used and limits the universality of the architecture for matrix computations of different sparsity levels.
- **Physical implementation challenges or constrained repairability:** The designs of using redundant PEs to replace faulty ones to achieve hardware repairs do not change the computational structure of the hardware architecture and have a certain universality [14]–[16]. However, these methods may face challenges such as excessive routing resources leading to large area overheads, impacting the overall physical implementation of the architecture. Additionally, when faults are concentrated in local areas, the repair capabilities of this method decrease, and in some cases, the repair mechanisms may even fail.

We aim for a highly adaptable matrix multiplication architecture with robust repair capabilities. It should handle various matrix multiplication operations without extra software overhead or algorithmic constraints, such as sparse preprocessing, and maintain computational throughput without relying on specific fault distributions. In this paper, we propose GEMMHeal, an efficient self-repairing systolic array-like architecture for matrix multiplication. We propose a high-throughput error correction (EC) core in a parallel computing style as a redundant computation unit, decoupled from the conventional matrix multiplication array. This allows the EC core's resources to be fully utilized when a fault occurs in the matrix multiplication array, preventing performance loss caused by imbalanced resource utilization in scenarios such as faults concentrated in a small localized area. Additionally, the EC core incorporates redundancy to enable self-repair, enhancing its robustness against circuit faults. The conventional matrix multiplication array is optimized based on the systolic array architecture by eliminating data dependencies between PEs. This allows the workload of faulty PEs to be entirely offloaded to the EC cores without incurring additional delays or scheduling overhead. We implement the proposed efficient self-repair systolic array-like architecture at the back-end. Physical implementation results indicate that the proposed architecture is hardware-friendly, with minimal additional area overhead and no significant congestion in critical paths or back-end layout and routing. Specifically, our contributions are as follows:

- We propose a high-throughput error correction (EC) core architecture with redundancy, which can effectively mitigate performance loss due to PE failures and is resilient against potential faults within the EC core.

* Equal Contribution
Corresponding author: yangfan@fudan.edu.cn

- We optimize the conventional matrix multiplication array and decouple it from the error correction core, ensuring that the coverage or resource utilization of the error correction core is not constrained by the spatial locality of faults.
- We perform back-end implementation to validate the reliability of this architecture, demonstrating that it incurs minimal additional area overhead and does not introduce significant physical implementation challenges.

The remainder of this paper is organized as follows: Sec. II introduces the matrix multiplication algorithm and existing repair methods for systolic arrays. Sec. III details the architecture and design of our proposed solution. In Sec. IV, we present a performance evaluation of our proposed design, including experiments on the redundancy of the EC core and comparisons with other existing works. Finally, Sec. V concludes our work.

II. BACKGROUND

A. Matrix Multiplication

In matrix multiplication, the prevalent techniques are the inner product method [17]–[19] and the outer product method [20]–[22]. The inner product multiplies each row of one matrix with the corresponding column of another matrix, summing the results to form an output matrix element. The outer product multiplies the vectors to produce intermediate matrices, which are summed to get the final matrix. These methods differ in their parallelism and data flow in hardware. In conventional systolic arrays, an efficient architecture designed for matrix multiplication, data accumulates through inner-product computations as products of weights and features flow between tightly connected PEs. This setup results in high computational density and throughput at a relatively low cost, making systolic arrays widely used.

Systolic arrays handle specific matrix sizes well and are adaptable to smaller matrices with padding. For larger matrices, a common approach is to use block matrix multiplication, where submatrices (blocks) are treated as elements of the original matrix [23]. The systolic array then multiplies these submatrices in sequence, combining or summing the resulting submatrices to form the final result matrix. This strategy allows systolic arrays to handle larger matrix multiplication tasks efficiently without increasing hardware costs or risking the underutilization of larger systolic arrays.

B. Repair Methods for Systolic Array

However, due to the large and dense number of PEs in a systolic array, combined with data dependencies between them, this architecture is highly sensitive to faults. As large-scale computational tasks require larger or more systolic arrays to increase throughput, the likelihood of hardware failures also increases. In this context, the primary concern is the failure of PEs within the array. Therefore, effective detection and repair of faults in such array structures is crucial.

Existing methods can be broadly classified into two categories: redesigning algorithms or disabling faulty units in hardware to avoid their use, and introducing redundant units to take over the computational tasks of the faulty ones [24]. In our view, algorithmic optimization can only address specific issues and lacks broader applicability, while disabling faulty units in hardware can lead to complex data scheduling or waste of computational resources. We prefer to use redundant units for repairs, but a key challenge lies in how to allocate these units for optimal effectiveness. Furthermore, there are three specific requirements to consider:

1) The physical placement of redundant units is a critical factor to consider. The computational efficiency of the existing resources should not be compromised by the insertion of redundant units, such as being restricted by routing or layout limitations.
2) We also aim for the redundant units to provide the best repair results with minimal additional cost.
3) The redundant units should possess repair capabilities that are not constrained by spatial limitations, meaning they should be able to address faults occurring anywhere across the system.

Previous works may not have met the three fundamental requirements mentioned above. Kim [14] and Cho [15] do not satisfy the first and second requirements, and Lee [16] does not address the third.

III. PROPOSED DESIGN

In this section, we first introduce the overall architecture of GEMMHeal, which supports efficient matrix multiplication and can repair faulty PEs at any location without affecting computation speed. We then detail our optimized design, which decouples matrix multiplication from the error correction core. Finally, we describe the optimized data flow of the decoupled matrix multiplication and repair processes, as well as the detailed design of the error correction core.

A. Overall Architecture

The architecture of GEMMHeal consists of four main components: the feature buffer and weight buffer as input buffers, the output buffer, the systolic array-like matrix multiplication module, and the error correction (EC) core. Fig. 1 illustrates the overall architecture of GEMMHeal. The weight and feature buffers provide input data to the matrix multiplication array and the EC core, respectively, with results from both modules merged in the output buffer.

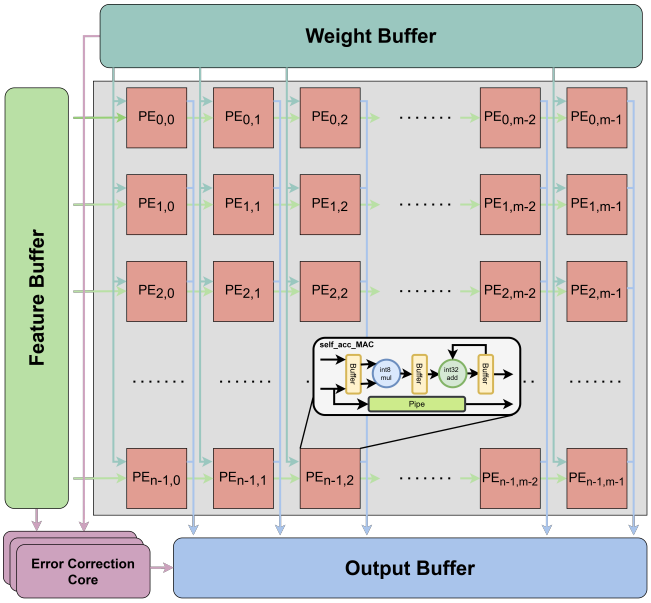

Fig. 1: The Overall Architecture of GEMMHeal.We use an n × m PE array as a case.

When some PEs in the matrix multiplication array encounter faults, we utilize the EC core to handle the workload initially assigned to these faulty PEs. The process involves first creating a fault table to record the coordinates of the faulty PEs. Based on these coordinates, we select the relevant input data from the weight and feature buffers and feed it into the EC core for computation. The results are then placed in specific positions within the output buffer according to the faulty PE coordinates, thus achieving effective fault recovery within the matrix multiplication array.

B. Optimized Matrix Multiplication Array

In our design of the matrix multiplication array, we introduce an optimized approach based on the conventional systolic array architecture. In the systolic array, each PE depends on data from neighboring PEs, with values accumulating along a propagation chain. This dependency creates a challenge: incorrect computation results due to any faulty PE propagating to all subsequent PEs. In this case, we must not only complete the computation task for this PE but also correctly propagate the data to neighboring PEs without affecting the pipeline rhythm. There may be a significant area overhead and physical implementation challenge because the fault may occur at any location in the array, and we need to provide replacement PEs for each location.

We observe that in matrix multiplication using the outer product approach, the data dependencies are significantly reduced. Based on this, we propose a systolic array-like architecture built on the outer product algorithm, which decouples the timing dependencies between PEs, allowing each PE to compute elements of the result matrix independently. Specifically, we substitute inter-PE accumulation with self-accumulation within each PE. Each PE independently receives broadcasts from the weight buffer and sequentially propagates feature data. Upon completion of the matrix computation, the output buffer incrementally collects the outputs from each column of PEs to produce the result matrix. Fig. 1 illustrates the microarchitecture of PE and the data flow through the array.

In addition to decoupling the PE units from one another, this optimized matrix multiplication array provides an additional benefit: it also decouples the EC core from the matrix multiplication array. With the EC core and the matrix array operating as separate data paths, the EC core can handle tasks assigned to faulty PEs without needing to manage data dependencies within the computational flow of the matrix array. This independence also allows the EC core to avoid the spatial locality constraints typically associated with faulty elements. In Sec. III-C, we explain how our design avoids timing issues that could arise during the computation process.

C. Data Flow Optimization

The weight and feature matrices need to be fed separately into the matrix multiplication array and the EC core for computation, with each requiring different data. The matrix multiplication array, as the core unit performing matrix multiplication, requires the entire data set, whereas the EC core only needs a limited subset: specifically, the data required by faulty PEs. The EC core is responsible for completing the calculations originally assigned to these faulty PEs.

However, there is a challenge. Faults can occur at any position within the array, and when a faulty PE is positioned further down the data propagation chain, the required input data arrives later in the calculation pathway. Since result data is output simultaneously, a fault in a downstream PE means that the EC core has even less time available to complete the necessary calculations. If the EC core does not have sufficient computation time, it risks back-pressuring the matrix output and reducing overall computational throughput.

We address this issue significantly by optimizing the input data flow. As shown in Fig. 2, each buffer bank's data port is widened, allowing data to be temporarily held in the dispatcher upon read access. These dispatchers are responsible for scheduling and sequentially outputting data to the matrix multiplication array. The EC core is given higher priority in selecting data from these dispatchers. Based on the fault table that records the coordinates of faulty PEs, the EC core retrieves the required feature or weight data and starts processing as soon as matrix multiplication begins.

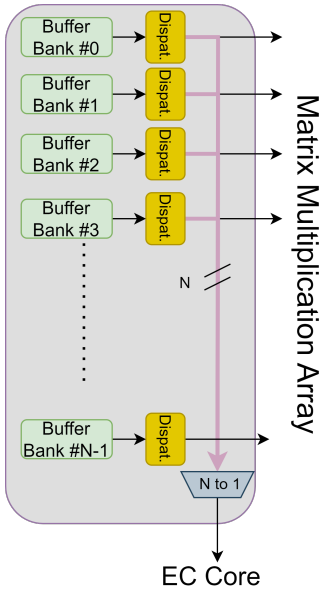

Fig. 2: The Optimized Buffer. It can offer flexible scheduling for concurrent delivery of weights/feature data to the EC core and matrix multiplication arrays.

This setup allows the EC core to begin repairing faulty PEs as early as possible, minimizing dependency on faulty PE positions within the matrix and ensuring that repairs do not delay overall matrix computation. When the EC core completes its calculations, it sends the data to the output buffer for storage. This process is independent of the matrix multiplication calculation, with specific output buffer locations determined by the fault table. As long as the fault rate within the matrix multiplication array remains below a certain threshold, this optimized data flow successfully mitigates any adverse effects on overall computational throughput.

D. Error Correction Core

The error correction (EC) core receives the input data originally intended for the faulty PE, completes the workload the faulty PE is responsible for, and sends the results back to the output buffer. We optimize the EC core's throughput to prevent bottlenecks in the throughput of the matrix multiplication array.

Similar to previous works, we also employ a fundamental multiply-accumulate flow to handle the workload of faulty PEs. However, unlike prior approaches that replace PEs on a one-to-one basis, which required identical PEs to substitute faulty ones while restructuring data flow to maintain timing and dependency, our approach removes these complexities. For example, Lee [13] assigns each repair unit to specific matrix multiplication columns, activating only when a specific column encounters issues. In our design, by offloading faulty PE tasks to EC Cores, we eliminate the need for the more constrained and lower-covered one-to-one PE replacement method.

As shown in Fig. 3, we demonstrate the microarchitecture of an EC core with a parallelism degree of 4. In GEMMHeal, we use a multiply-add tree structure to complete the multiply-accumulate flow in the EC core, improving its computational parallelism. The EC core allows the original multiple-cycle multiplication task to be completed in one cycle and finally summed up by an addition tree to obtain the partial sum, which requires less latency to obtain the original output of the faulty PEs. This design also enables the EC

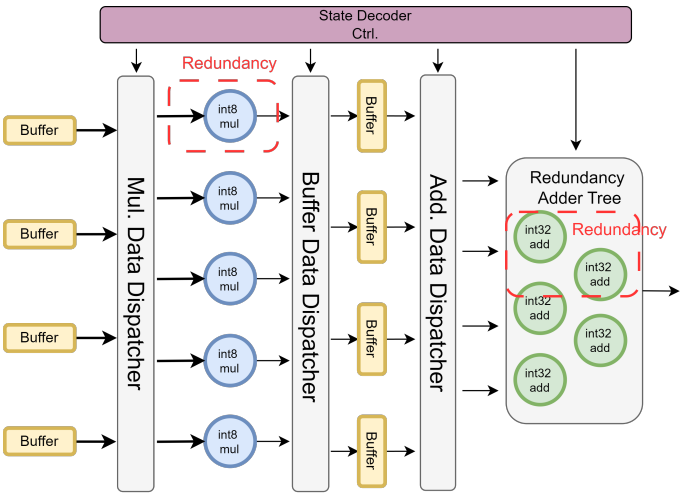

Fig. 3: Microarchitecture of the Error Correction (EC) Core with Parallelization and Redundancy Design. Here, a parallelism degree of 4 is shown as an example. We believe that the choice of these parameters for the EC core should balance both efficiency and area.

core to handle multiple faulty PEs sequentially, achieving a high error correction coverage rate. When the fault rate is within the EC core's maximum process threshold, the EC core can complete correction tasks in advance, ensuring no impact on the matrix multiplication array's efficiency.

Additionally, we address the spatial locality limitations of fault tolerance in prior designs. In Lee's work [13], if faulty PEs are concentrated in two specific columns, only the corresponding repair units assigned to these columns would be active, leaving other repair units idle—reducing resource utilization and constraining computational throughput. By integrating an optimized matrix multiplication array with our proposed EC core architecture, we eliminate spatial locality constraints on fault occurrence. This approach also minimizes physical implementation challenges. Experimental data is presented in Sec. IV-C to demonstrate the robustness of our proposed self-repair architecture in handling various fault distributions.

Unlike previous works, we account for the possibility of faults occurring in any computation cell within the architecture, as MACs in both the matrix multiplication array and the EC core exhibit nearly identical fault probabilities. To improve fault coverage further, we introduce redundancy in the EC core's design, incorporating extra multiplier and adder units, as illustrated in Fig. 3. When a fault occurs in the EC core, priority state encoding is applied to the faulty multipliers or adders. Based on these encoded states, the controller generates control signals to direct the workload to functional units. This redundancy improves the fault tolerance of the EC core and enhances the robustness of the overall design. Notably, this redundancy requires minimal additional resources, providing reliable error correction capabilities within GEMMHeal. Excessive resource overhead for redundancy design is also discouraged, as it primarily serves as a backup. Ideally, the additional resources for redundancy in the EC core should align with the EC core's overhead relative to the matrix multiplication array, ensuring balanced, efficient self-repair.

IV. Evaluation

A. Methodology

In this section, we present the experimental setup and performance evaluation for GEMMHeal. We implement GEMMHeal in Verilog HDL and conduct physical implementation with a 32nm technology library [25]. The specific experimental configuration is shown in Tab. I: the matrix multiplication array in GEMMHeal is scaled to 16×16, with each EC core having a parallelism degree of 16 and a redundancy degree of 4.

TABLE I: Experiment Configuration

Process Library		32nm
Synthesis Tool		Design Complier
Physical Design Tool		IC Compiler
Parameter	Matrix Multiplication Array	Size: 16×16
	Error Correction Core	Size: 16 Redundancy: 4

B. Feasibility Analysis

To validate the practical feasibility of our proposed architecture, we configure GEMMHeal with an EC core using the parameters and tools listed in Tab. I. Fig. 4 shows the back-end layout of GEMMHeal, with the orange section highlighting the EC core. Additionally, we present detailed area data for GEMMHeal and compare it with a baseline design, as detailed in Tab. II. The difference between GEMMHeal and the baseline lies in the inclusion of our proposed EC Core in GEMMHeal, while the rest of the hardware configuration remains identical.

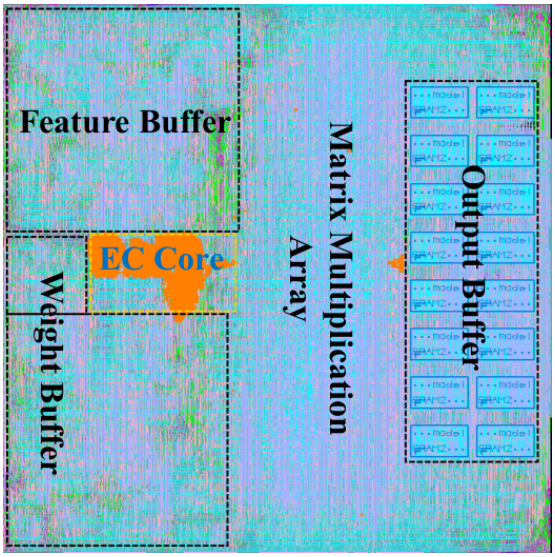

Fig. 4: Physical Design of our Proposed Design based on 32nm Process.

As shown in Tab. II, the addition of the EC core slightly reduces GEMMHeal's operating frequency from 695 MHz to 690 MHz, a nearly negligible difference. This indicates that the inclusion of the EC core has minimal impact on the overall floorplan and routing complexity, demonstrating that the EC core is decoupled from the matrix multiplication array. In terms of area, GEMMHeal with a single EC core occupies approximately 1.11 mm^2, with the EC core

TABLE II: Area Comparison of Baseline and GEMMHeal

Design		Baseline	GEMMHeal
Frequency		695 MHz	690 MHz
Area (mm^2)	Feature Buffer	0.18 (16.6%)	0.18 (16.3%)
	Weight Buffer	0.20 (18.3%)	0.20 (18.0%)
	Output Buffer	0.25 (22.7%)	0.25 (22.7%)
	PE Array	0.41 (38.1%)	0.41 (36.6%)
	EC Core	-	**0.02 (1.8%)**
	Total	1.08 (100%)	1.11 (100%)

occupying 0.02 mm^2, or about 1.8% of the total area. The additional area required by the EC core is minimal. Even when focusing solely on the matrix multiplication array's area, this area increase remains under 5%.

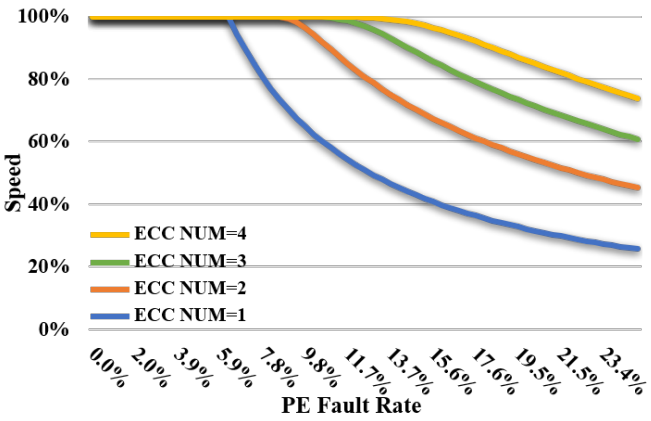

Fig. 5: Computation Speed of Matrix Mutiplication with Varying Numbers of Non-Redundant EC Cores at Different PE Fault Rates.

C. Experimental Result

Fig. 5 illustrates the computation speed of GEMMHeal under full workload at varying PE fault rates with different numbers of EC cores. We assume no faults occur within the EC core. The most immediate observation is that, as the number of EC cores increases, the fault tolerance of GEMMHeal improves, allowing it to maintain consistent computational speed even under increasingly adverse conditions—namely, with a higher number of faulty PEs. This outcome is intuitively expected. Specifically, GEMMHeal equipped with four EC cores can sustain matrix multiplication performance even with a PE fault rate as high as 8.2%, and at a 25% PE fault rate, it still retains approximately 74% of its computational speed.

TABLE III: Comparison of Repair Rate within Different Works (Faulty PE: 16).

Fault Area	32x32	64x64	128x128
One-D [14]	0.0117	0.1307	0.3802
Two-D [15]	0.4210	0.8732	0.9801
STARIT [13]	0.0596	0.5449	0.8561
AESAR [16]	0.6453	0.9470	0.9933
GEMMHeal	**1**	**1**	**1**

It is noteworthy that previous works [13]–[16], often assess a metric known as the "Faulty Area", which indicates the spatial concentration of faulty PEs. When the Faulty Area is smaller, the faulty PEs are more spatially concentrated, and the chance of multiple faults in the same repair group becomes larger. Redundant PEs in the same repair group have difficulty in replacing or repairing multiple faulty PEs, resulting in a lower repair rate. We compile the repair rates from previous works under various Faulty Areas and compare them with our design, which is shown in Tab. III. The repair rates of previous works decline significantly as the Faulty Area decreases, indicating that the repair effectiveness is constrained by the spatial locality of faults. For instance, when faults occur within a 32×32 region, the repair rate of One-D [15] is only 42.10%, significantly lower than the 98.01% repair rate achieved when the same number of faults is spread over a larger 256×256 region. Although the probability of failure is roughly equal for each PE, localized clusters of faults—though less likely—would almost always cause these architectures to fail rapidly. This failure occurs because the repair units, which are evenly distributed across the functional PEs, follow a region-based array assignment strategy. As a result, when faults concentrate within smaller array regions, the repair units in the affected areas become overloaded, while those in other areas remain idle.

In GEMMHeal, however, the decoupling of the matrix multiplication array and EC core allows for a repair rate that is unaffected by the spatial locality of faults. Thus, GEMMHeal achieves consistent repair rates across different fault distributions. In certain corner cases, our proposed architecture performs significantly better than previous designs. Due to the decoupling of EC core and PE arrays in our architecture, our repair rate is 100% when the number of failed PEs does not exceed the size of the EC core.

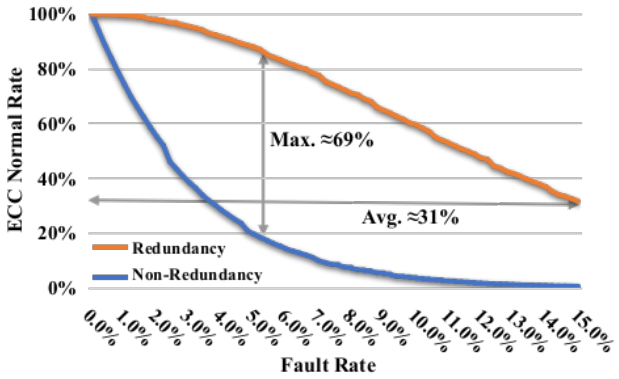

Fig. 6: Comparison of Normal Rates: Redundant vs. Non-Redundant EC Cores Under Varying PE Fault Rates.

To further evaluate the EC core architecture, we conduct more rigorous experiments and a deeper analysis. Previous works often considered only the probability of faults in the matrix multiplication array, overlooking the fact that EC cores themselves have a non-negligible fault rate, approximately equal to that of the array. We define the probability of the EC core functioning correctly as the "Normal rate". This rate is directly related to the correction capability of the core and significantly impacts the computational throughput of matrix multiplication. Therefore, we propose a self-repairable EC core architecture and assess its performance under this consideration. Fig. 5 illustrates EC core performance under ideal conditions. Our proposed solution includes a redundancy design for the EC core, which effectively mitigates performance degradation caused by EC core failures. This redundancy setup is more resource-efficient, as it functions similarly to a backup for the core.

In our experiments, we account for PE failures within the EC core itself. Fig. 6 shows the impact on the EC core's normal rate under various PE fault rates when the redundancy-style design is applied, compared with a non-redundant EC core. Notably, the redundant EC core significantly mitigates the degradation of the normal rate. When the fault rate of the EC core is approximately 5.2%, the redundant design can achieve a maximum increase of 69% in the normal rate compared to the non-redundant version. Assuming a maximum fault rate of 15% for a relatively severe PE fault, we evaluate the impact of our redundancy design on the EC core's normal rate when the PE fault rate is within this threshold. Compared to the non-redundant version, our redundancy design for the EC core achieves an average 31% increase in the normal rate. Here, we define each adder or multiplier as a minimal unit, with a single failure in either unit may fail the EC core. Due to this stringent assumption, the normal rate for the EC core in Fig. 6 appears to degrade more rapidly as the PE fault rate increases.

V. CONCLUSION

In this paper, we introduce GEMMHeal, a highly efficient self-repairing architecture optimized for matrix multiplication in machine-learning models like Transformers. GEMMHeal decouples the error correction core by optimizing the matrix multiplication array, data flow, and buffer design to ensure robust performance and efficient resource use under faults. By leveraging the redundancy within the error correction core, the error correction core in GEMMHeal can perform limited self-repair, enhancing its fault tolerance. We conduct evaluations of GEMMHeal using a 32nm process, demonstrating its physical feasibility. GEMMHeal is expected to become a robust solution for matrix multiplication architectures.

ACKNOWLEDGMENTS

This work was supported by the Science and Technology Commission of Shanghai Municipality under Project 24BC3201000. The authors would like to thank the reviewers for their valuable feedback.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[2] A. Ivanov, N. Dryden, T. Ben-Nun, S. Li, and T. Hoefler, "Data movement is all you need: A case study on optimizing transformers," 2021. [Online]. Available: https://arxiv.org/abs/2007.00072

[3] J. Liu, G. Dai, H. Xia, L. Guo, X. Shi, J. Xu *et al.*, "Tstc: Two-level sparsity tensor core enabling both algorithm flexibility and hardware efficiency," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, Oct 2023, pp. 1–9.

[4] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu, and P. Micikevicius, "Accelerating sparse deep neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2104.08378

[5] J. Yu, Y. Fan, H. Wang, Y. Qiao, Z. Wu, X. Xiong, X. Yao, H. Yao, and Y. Zhang, "Fullsparse: A sparse-aware gemm accelerator with online sparsity prediction," in *Proceedings of the 21st ACM International Conference on Computing Frontiers*, ser. CF '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 298–301. [Online]. Available: https://doi.org/10.1145/3649153.3649180

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

[7] M. Traiola, F. F. dos Santos, P. Rech, C. Cazzaniga, O. Sentieys, and A. Kritikakou, "Impact of high-level synthesis on reliability of artificial neural network hardware accelerators," *IEEE Transactions on Nuclear Science*, vol. 71, no. 4, pp. 845–853, 2024.

[8] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17 322–17 341, 2017.

[9] S. Kundu, S. Banerjee, A. Raha, S. Natarajan, and K. Basu, "Toward functional safety of systolic array-based deep learning hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 3, pp. 485–498, 2021.

[10] J. Park, M. Naumov, P. Basu, S. Deng, A. Kalaiah, D. Khudia, J. Law, P. Malani, A. Malevich, S. Nadathur, J. Pino, M. Schatz, A. Sidorov, V. Sivakumar, A. Tulloch, X. Wang, Y. Wu, H. Yuen, U. Diril, D. Dzhulgakov, K. Hazelwood, B. Jia, Y. Jia, L. Qiao, V. Rao, N. Rotem, S. Yoo, and M. Smelyanskiy, "Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications," 2018. [Online]. Available: https://arxiv.org/abs/1811.09886

[11] J. J. Zhang, K. Basu, and S. Garg, "Fault-tolerant systolic array based accelerators for deep neural network execution," *IEEE Design Test*, vol. 36, no. 5, pp. 44–53, 2019.

[12] M. Sadi and U. Guin, "Test and yield loss reduction of ai and deep learning accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 1, pp. 104–115, 2022.

[13] H. Lee, J. Kim, J. Park, and S. Kang, "Strait: Self-test and self-recovery for ai accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 9, pp. 3092–3104, 2023.

[14] J. Kim and S. Reddy, "On the design of fault-tolerant two-dimensional systolic arrays for yield enhancement," *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 515–525, 1989.

[15] K. Cho, I. Lee, H. Lim, and S. Kang, "Efficient systolic-array redundancy architecture for offline/online repair," *Electronics*, vol. 9, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/2/338

[16] H. Lee, J. Park, and S. Kang, "An area-efficient systolic array redundancy architecture for reliable ai accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 10, pp. 1950–1954, 2024.

[17] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan *et al.*, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Feb 2020, pp. 58–70.

[18] H. Nair, P. Vellaisamy, A. Chen, J. Finn, A. Li, M. Trivedi, and J. P. Shen, "tugemm: Area-power-efficient temporal unary gemm architecture for low-precision edge ai," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[19] E. Taka, D. Gourounas, A. Gerstlauer, D. Marculescu, and A. Arora, "Efficient approaches for gemm acceleration on leading ai-optimized fpgas," in *2024 IEEE 32nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2024, pp. 54–65.

[20] S. Pal, J. Beaumont, D.-H. Park, A. Amarnath, S. Feng *et al.*, "Outerspace: An outer product based sparse matrix multiplication accelerator," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Feb 2018, pp. 724–736.

[21] R. Hojabr, A. Sedaghati, A. Sharifian, A. Khonsari, and A. Shriraman, "Spaghetti: Streaming accelerators for highly sparse gemm on fpgas," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 84–96.

[22] N. G, N. S, and K. S, "Moscon: Modified outer product based sparse matrix-matrix multiplication accelerator with configurable tiles," in *2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID)*, 2023, pp. 264–269.

[23] Y. Wang, C. Zhang, Z. Xie, C. Guo, Y. Liu, and J. Leng, "Dual-side sparse tensor core," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 1083–1095.

[24] I. Oz and S. Arslan, "A survey on multithreading alternatives for soft error fault tolerance," vol. 52, no. 2, Mar. 2019. [Online]. Available: https://doi.org/10.1145/3302255

[25] Synopsys, "Synopsys educational programs-saed," urlhttps://www.synopsys.com/company/contact-synopsys/office-locations/armenia/educational-programs.html, accessed on 2024-10-26.