Fault Tolerance Techniques for Error Detection and Correction in Systolic Arrays

Selvaganesh M
Department of ECE,
Sri Ramakrishna Engineering College,
Coimbatore, India
selvaganesh.m@srec.ac.in

Abinaya R
Department of ECE,
Sri Ramakrishna Engineering College,
Coimbatore, India
abinaya.2102003@srec.ac.in

Bala Amuthan S
Department of ECE,
Sri Ramakrishna Engineering College,
Coimbatore, India
balaamuthan.2102021@srec.ac.in

Balaji S
Department of ECE,
Sri Ramakrishna Engineering College,
Coimbatore, India
balaji.2102022@srec.ac.in

***Abstract*—This paper presents an efficient approach for enhancing the reliability of matrix computations performed on systolic array architectures through matrix code-based error detection and correction techniques. With the increasing deployment of systolic arrays in high-performance and real-time applications, fault tolerance has become a critical requirement to maintain data integrity in the presence of hardware errors. The proposed method leverages structured redundancy and algorithmic checks to identify and correct faults with minimal impact on computation speed and resource usage. By reviewing existing fault-tolerant strategies and integrating matrix coding principles, this work offers a practical solution that improves error resilience without significantly increasing complexity. Simulation results and analysis demonstrate the effectiveness of the approach in maintaining accuracy under fault conditions, making it suitable for applications in scientific computing, AI accelerators, and embedded systems.**

Keywords—matrix computation, systolic array, error detection, hamming code, fault - tolerance

I. INTRODUCTION

A key operation in computer science, matrix multiplication (MxM) supports a wide range of applications, including computer vision, radio signal processing, financial market analysis, and weather prediction. Due to the need for mission-critical dependability and real-time operation, several of these industries have extremely high requirements for fault tolerance and computing efficiency. However, matrix computing systems themselves are susceptible to radiation-related defects, notably single and multi-bit upsets, which can manifest as either an application-level problem or a whole system-level disaster. Redundancy and advanced error correction codes (ECCs) are typically used as traditional defences against these vulnerabilities, despite the fact that they significantly increase system latencies and hardware overheads. This work addresses these problems by introducing a novel paradigm for error detection and correction for MxM that drastically lowers algorithmic and architectural costs, which are an order of magnitude higher than those of a polynomial. Our approach improves efficiency by combining advanced matrix codes for memory protection, a divide-symbol technique for maintaining reliability with negligible latency, and an encoder-reuse strategy for reducing circuit space without compromising encoding and decoding integrity.

II. LITERATURE SURVEY

For systolic array-based matrix multiplication, Lu, Su, and Huang suggest a fault-tolerant method that improves reliability even when there are several malfunctioning processing elements (PEs). Their approach uses a pair-matching process in which a fault-free PE is designated to serve as a proxy for each problematic PE, performing the required calculations. This method reduces circuit area and increases fault tolerance by doing away with the requirement for a fault-free column, which was present in earlier approaches. To strike a compromise between hardware overhead and fault tolerance capabilities, the authors present two pair-matching algorithms: one-dimensional and two-dimensional. According to experimental data, this approach is appropriate for applications where system reliability is crucial since it not only increases fault tolerance but also makes controller and PE design simpler. This paper introduces two pair-matching schemes: row-based pair-matching and column-based pair-matching [1]. Employing two-dimensional pair-matching offers higher fault tolerance compared to using one-dimensional pair-matching, but it also increases the controller area[1]. Their method is two-step: first, fault-free PEs carry out their assigned calculations; then, they carry out the jobs of their associated faulty PEs. Implemented using a TSMC 40nm cell library, the approach displays better fault resilience and reduced circuit area, making it effective for scenarios with a higher number of faulty PEs.

The authors, Y. Wang, Y. Chen, and H. Zhou of this work discuss the growing demand for hardware accelerators to be reliable, particularly in systolic array processors, which are frequently employed for high-speed matrix calculations in fields such as scientific applications, machine learning, and signal processing. As these systems become more complicated and scale, errors—whether brought on by ageing, environmental noise, or hardware flaws—can have a big influence on the accuracy of the system. The authors address this by proposing a fault-tolerant systolic array architecture, which is tested and implemented with FPGA technology. The paper's main contribution is the creation of a

systolic array processor that can maintain accuracy even when processing elements (PEs) malfunction.

Wu et al. provide FT-GEMM, a high-performance and fault-tolerant general matrix-matrix multiplication (GEMM) implementation designed for x86 CPUs. By combining memory-intensive operations into GEMM assembly kernels, FT-GEMM incorporates fault-tolerant functionality at the algorithmic level, addressing the vulnerability of large-scale computing platforms to soft errors. With little performance overhead, this approach guarantees real-time error detection and correction. A cache-friendly parallel execution technique is also presented by the authors. In addition to maintaining excellent reliability under frequent error injections, experimental assessments on Intel Cascade Lake processors show that FT-GEMM outperforms existing libraries such as Intel MKL, OpenBLAS, and BLIS by 3.5% to 22.14% in both serial and parallel GEMM operations. FT-GEMM, a high-performance GEMM being capable of tolerating soft errors on-the-fly[3].

S. Zhang and K. Roy propose a low-overhead error detection method tailored for systolic array-based matrix multiplication accelerators, which are widely used in deep learning and high-performance computing. Recognizing that traditional fault-tolerant techniques often incur significant resource and performance costs, the authors introduce a lightweight scheme that detects faults during computation with minimal hardware complexity. Their approach relies on duplicating only selected computations and comparing results to flag inconsistencies, rather than relying on full redundancy. This selective strategy significantly reduces power and area overhead while maintaining high fault coverage. Through simulation and hardware evaluation, the paper demonstrates that the proposed method effectively detects errors with very limited performance impact, making it highly suitable for energy-efficient and reliable computing in modern AI accelerators. the work offers a scalable and effective way to improve systolic array fault tolerance, resulting in more durable and dependable accelerators for high-performance computing. Systems that need both a lot of processing power and little overhead, such AI accelerators and embedded systems, benefit greatly from this strategy.

AxSA is a unique design for approximation systoli c arrays that improves matrix multiplication operations' perf ormance and power consumption, according to a work by W aris et al.By optimising the systolic arrays' critical path time, the scientists were able to achieve significant reductions of up to 32% for 8-bit operations and 64% for 32bit operations. This enhancement is made while also lowering the system's overall power consumption.Two architectural designs, Ax1 and Ax2, are presented; Ax2 is clearly the more power efficient design, providing an impressive 28% power reducti on over traditional systolic arrays.According to the Structural Similarity Index (SSIM), Ax2 only loses 5% of its accuracy despite these energy savings, indicating that the loss of computational precision is negligible and appropriate for many applications that do not require precise answers. Because of the study's obvious trade-off between accuracy, speed, and energy efficiency, AxSA is a perfect fit for applications where approximation is not only acceptable but also advantageous for system performance as a whole. In the context of contemporary, energy-conscious computing systems, this research is especially pertinent because the authors show how approximate computing techniques can be incorporated into systolic array designs to produce power-efficient hardware for signal processing and matrix multiplication tasks.

III. PROPOSED METHOD

By integrating a hybrid error correcting method with Hamming and parity codes and Lightweight Algorithm-Based Fault Tolerance (LABFT) for real-time fault detection, the proposed method enhances the reliability of systolic array-based matrix multiplication. Tested through Verilog HDL for fault injection testing, the method ensures efficient fault detection and correction without any performance loss. For mission-critical operations, it presents a low-overhead, high-performance solution for fault-tolerant systolic array topologies

METHODOLOGY

1. LIGHT ABFT METHOD:

A resource-efficient fault detection method called Light ABFT (Lightweight Algorithm-Based Fault Tolerance) was created for matrix operations in systolic arrays, Light ABFT minimises overhead by using a reduced checksum methodology, in contrast to typical ABFT algorithms that demand significant processing resources and additional storage. Prior to multiplication, it calculates the row checksum of Matrix B and the column checksum of Matrix A, producing the anticipated output checksum. After executing matrix multiplication, the row and column sums of the resulting Matrix C are compared with the expected checksums. Without having a major influence on system performance or resource utilisation, a mismatch indicates a computation error, allowing for the rapid and efficient diagnosis of hardware faults.
The steps are as follows:

A. INPUT CHECK SUM GENERATION

In this step, checksums of the input matrices are calculated to serve as a reference for verifying the correctness of the output:

- For Matrix A: Compute the column-wise checksum
  $CA[j]=i=1\sum m\ A[i][j]$
  This results in a $1 \times n$ vector.
- For Matrix B: Compute the row-wise checksum
  $RB[i]=j=1\sum p\ B[i][j]$
  This results in an $n \times 1$ vector.

These values represent the expected behavior of input data.

B. COMPUTE EXPECTED OUTPUT

Using the above input checksums to calculate a reference checksum that represents the expected behavior of the final result:

Expected Checksum= CA*RB= $j=1\sum n$ CA[j]·RB[j]

This produces a single scalar value which acts as the predicted total sum of the output matrix if no fault occurs during multiplication.

C. COMPUTE MATRIX MULTIPLICATION

The actual matrix multiplication is performed using hardware such as a systolic array, which efficiently processes matrix elements in a pipelined and parallel fashion:

- C[i][j]= $k=1\sum n$ A[i][k]·B[k][j]

This yields the output matrix C of size m×p.

D. OUTPUT CHECKSUM CALCULATION

After computing Matrix C, generate the actual checksum by summing either:

- All elements of Matrix C
  ActualCheck sum= $i=1\sum m$ $j=1\sum p$ C[i][j]
  Or
- The row-wise or column-wise sum of C, depending on implementation. This computed checksum is the actual result of the matrix operation.

E. ERROR DETECTION

Now, compare the expected checksum (from Step 2) with the actual checksum (from Step 4):

- If they match: The computation is considered fault-free.
- If they don't match: An error has occurred—likely due to data corruption or hardware faults during matrix multiplication.

2. MATRIX CODES

a) DATA MATRIX FORMATION

Prepare the matrix MxM that contains the original data to be transmitted, stored, or processed.

$$M = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}$$

Fig 1 : Data matrix

b) APPLYING HAMMING CODES ON ROWS

Generate Hamming code by adding redundant bits that allow:

- Detection of up to 2-bit errors.
- Correction of 1-bit errors.

This transforms each row into a codeword.

Ri=[data bits+Hamming parity bits]

Now the matrix becomes wider,

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} & h_{14} & h_{15} \\ d_{21} & d_{22} & d_{23} & h_{24} & h_{25} \\ d_{31} & d_{32} & d_{33} & h_{34} & h_{35} \end{bmatrix}$$

Fig 2 : Matrix after applying hamming codes on rows

c) APPLY PARITY CODES ON COLUMNS

For each column, calculate the parity bit (even or odd) and add it as an additional row at the bottom:

pj=d1⊕d2⊕d3

Final matrix:

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} & h_{14} & h_{15} \\ d_{21} & d_{22} & d_{23} & h_{24} & h_{25} \\ d_{31} & d_{32} & d_{33} & h_{34} & h_{35} \\ p_1 & p_2 & p_3 & p_4 & p_5 \end{bmatrix}$$

Fig 3: Matrix formed after applying parity codes on columns

d) TRANSMISSION ON PROCESSING

Send or process the extended matrix. During this stage, an error (e.g., single-bit flip) might occur due to noise, soft faults, or hardware issues.

e) ERROR DETECTION AND CORRECTION

Once received:

- Use Hamming code (row-wise) to detect and correct any single-bit error within a row.
- If the Hamming check fails, it tells you which bit in the row is incorrect.
- Flip the bit to correct it.
- Use parity code (column-wise) to verify If a parity check fails, it signals an unrepaired error (e.g., if Hamming couldn't detect it due to multiple-bit errors).
- Helps in localizing errors further in conjunction with row data.

f) MATRIX RECOVERY

After correction and validation, the matrix is stripped of parity and Hamming bits, and the original data matrix is recovered.

3. HAMING CODES

- Hamming Code, developed in 1950 by Richard Hamming, is a basic error correction method that can detect double-bit and correct single-bit errors. It is widely applied in fields where accurate data transmission or storage is essential, including digital communications and memory systems. Its low complexity and efficiency make it suitable for hardware-based error correction.
- In order to monitor particular groups of data bits and enable the detection and correction of single-bit errors, the Hamming Code adds parity bits at powers of two positions. The formula 2 r ≥m+r+1, where r is the number of parity bits and m is the number of data bits, is used to determine the bare minimum amount of parity bits required.

IV. RESULTS AND DISCUSSION

Throughout testing and project development, the task was performed under a laboratory environment with simulation tools and industry-grade development tools. The primary hardware platform used for implementation and testing of the error correction and detection architecture was an efficient and adaptable board. VHDL/Verilog code deployment, synthesis, and simulation used toolchain tools such as the Vivado Design Suite. The simulator was critical in checking the logic design before hardware deployment. Testbenches were used to simulate various fault conditions, such as single-bit and multiple-bit faults, to check the behavior of the system and its fault-handling capability. These fault injection simulations allowed comprehensive testing of the LABFT and hybrid Hamming-parity mechanisms under a range of stress levels.

While testing hardware, the board was installed and monitored in real-time so that system operation and error correction response could be directly monitored. The environment also facilitated debugging with on-chip aids, like the Integrated Logic Analyzer (ILA), which enabled further insight into internal signal behavior under fault.

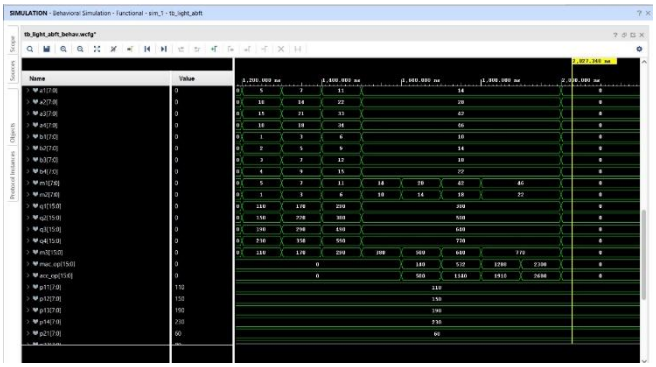

Fig.1 The Input Matrices A & B are Multiplied and Accumulated using Light ABFT method

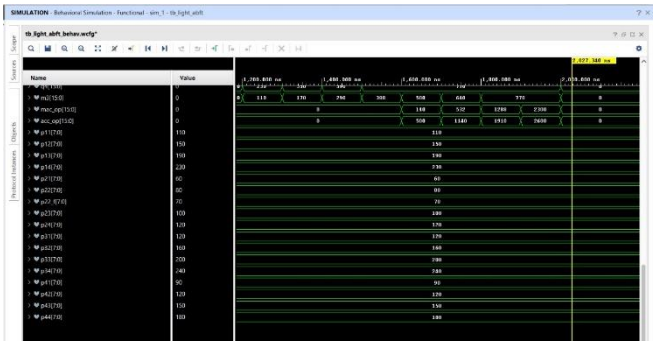

Fig.2 In p22 location Error is injected and the Output Matrix P is obtained using Light ABFT method

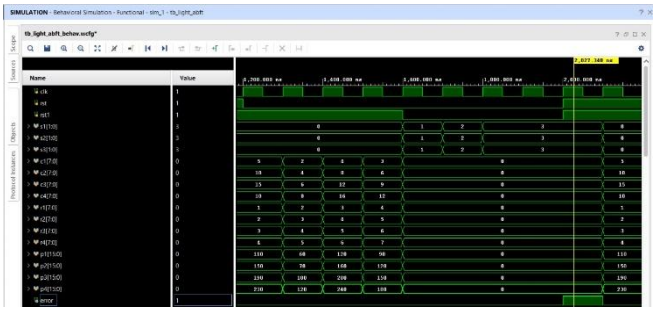

Fig.3 The Input Matrices and the Error injected Output Matrix are compared using Light ABFT method and Error is detected

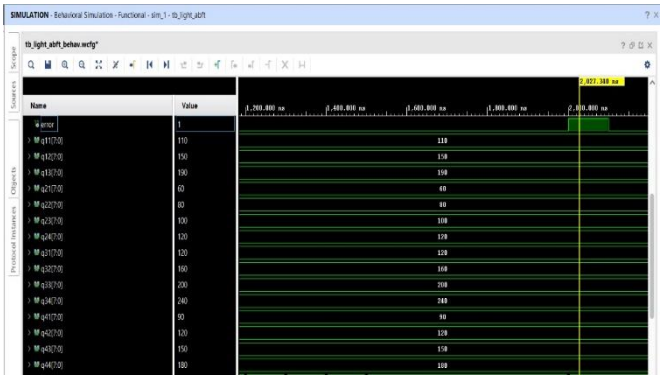

Fig.4 The Output Matrix P is corrected using Matrix Code which contains Hamming Code for rows and Parity Code for columns and obtains Matrix Q

ANALYSIS

TABLE 1: INPUT MATRICES

ERROR DETECTED IN OUTPUT MATRIX		ERROR CORRECTED OUTPUT MATRIX	
p_{11}	110	q_{11}	110
p_{12}	150	q_{12}	150
p_{13}	190	q_{13}	190
p_{14}	230	q_{14}	230
p_{21}	60	q_{21}	60
p_{22}_f	70	q_{22}	80
p_{23}	100	q_{23}	100
p_{24}	120	q_{24}	120
p_{31}	120	q_{31}	120
p_{32}	160	q_{32}	160
p_{33}	200	q_{33}	200
p_{34}	240	q_{34}	240
p_{41}	90	q_{41}	90
p_{42}	120	q_{42}	120
p_{43}	150	q_{43}	150
p_{44}	180	q_{44}	180

TABLE 2: ERROR DETECTED AND ERROR CORRECTED OUTPUT MATRICES

INPUT MATRIX OF A		INPUT MATRIX OF B	
a_{11}	5	b_{11}	1
a_{12}	10	b_{12}	2
a_{13}	15	b_{13}	3
a_{14}	10	b_{14}	4
a_{21}	2	b_{21}	2
a_{22}	4	b_{22}	3
a_{23}	6	b_{23}	4
a_{24}	8	b_{24}	5
a_{31}	4	b_{31}	3
a_{32}	8	b_{32}	4
a_{33}	12	b_{33}	5
a_{34}	16	b_{34}	6
a_{41}	3	b_{41}	4
a_{42}	6	b_{42}	5
a_{43}	9	b_{43}	6
a_{44}	12	b_{44}	7

V. CONCLUSION

The proposed structure effectively combines efficient correction procedures with lightweight real-time error detection in a fault-tolerant systolic array architecture for matrix multiplication. While the employment of parity and Hamming codes offers strong error correction across all dimensions of the computing matrix, the application of LABFT for fault detection guarantees low performance overhead. All things considered, the study shows a workable and trustworthy way to improve the reliability of matrix operations in systolic arrays, especially for crucial applications in signal processing and embedded systems.

VI. FUTURE WORK

Future revisions of this architecture can incorporate multi-bit error correction codes or adaptive fault tolerance mechanisms to accommodate more error rates and sophisticated fault models. Furthermore, incorporating machine learning algorithms can enhance fault pattern forecasting and facilitate dynamic corrective procedure adaptation. For improving performance and hosting applications in high-speed computing, AI accelerators, and edge devices with rigorous reliability demands, the design would be scaled to larger matrix size

REFERENCES

[1] Lu, Hsin-Chen, Liang-Ying Su, and Shih-Hsu Huang. "Highly Fault-Tolerant Systolic-Array-Based Matrix Multiplication." Electronics, vol. 13, no. 9, 2024.

[2] Y. Wang, Y. Chen, and H. Zhou, "Design and Implementation of a Fault-Tolerant Systolic Array Processor on FPGA," in *Proc. IEEE Int. Conf. Field-Programmable Technology (FPT)*, 2018, pp.

[3] S. Wu, Y. Zhai, J. Huang, Z. Jian, and Z. Chen, "FT-GEMM: A Fault Tolerant High Performance GEMM Implementation on x86 CPUs," presented at the Proceedings of the 28th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '23), Montreal, QC, Canada, Feb. 2023.

[4] S. Zhang and K. Roy, "A Low-Overhead Error Detection Scheme for Systolic Array-Based Matrix Multiplication Accelerators," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 29, no. 10, pp. 2671-2683, Oct. 2021.

[5] H. Waris, C. Wang, W. Liu, and F. Lombardi, "AxSA: On the Design of High-Performance and Power-Efficient Approximate Systolic Arrays for Matrix Multiplication," Journal of Signal Processing Systems, vol. 93, pp. 605–615, Jun. 2021.

[6] J. de Fine Licht, G. Kwasniewski, and T. Hoefler, "Flexible Communication Avoiding Matrix Multiplication on FPGA with High-Level Synthesis," in Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'20), Seaside, CA, USA, Feb. 2020.

[7] M. G. K. R. Reddy, V. R. Pudi, and K. L. Hsiao, "Matrix-Based Error Detection and Correction for Parallel Computations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 6, pp. 618-631, Jun. 2001.

[8] S. K. Gupta, M. G. Jafari, and M. R. Hashemi, "A Fault-Tolerant Algorithm for Matrix Multiplication Using Systolic Arrays," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 104-116, Jan. 2013.