FATAM: A FAULT-TOLERANTADDAPTIVE PPROXIMATE MULTIPLIER FOR ENERGY-EFFICIENT DEEP NEURAL NETWORK ACCELERATORS

[1]K.DURGA PRASAD, [2]Mr.P.VEERARAGHAVULU, [3]M. RAJESH, [4]V.SAI, [5]T. PAVAN KUMAR, [6]S. ASIF BASHA

[2]Assistant Professor, ECE Dept, RISE Krishna Sai Prakasam Group of Institutions, Valluru, AP

[1,3,4,5,6]Students, ECE Dept, RISE Krishna Sai Prakasam Group of Institutions, Valluru, AP

[1]karnatidurgaprasad2@gmail.com,[2]raghawa74@gmail.com,[3]mungamurirajesh123@gmail.com,[4]saivayala65@gmail.com,[5]pavankumar73336@gmail.com,[6]sayedasif545454@gmail.com

ABSTRACT

In a variety of safety-critical edge-AI applications with strict reliability, energy efficiency, and latency requirements, Deep Neural Network (DNN) hardware accelerators play a crucial role. Among the processing components of neural networks, multiplication consumes the highest hardware resources. In this research, a scalable adaptive fault-tolerant approximate multiplier (FaTAM) is proposed for ASIC-based DNN accelerators at both algorithm and circuit levels. The proposed adaptive approximate multiplier (AdAM) optimizes underutilized adder resources by employing an adaptive adder that utilizes an unconventional use of input Leading One Detector (LOD) values for fault detection. To further enhance hardware performance, a hybrid adder architecture and a gate-level optimized LOD design are incorporated into the multiplier structure. Additionally, a lightweight fault mitigation technique is introduced, which mitigates detected faults by forcing the identified erroneous bits to zero. The proposed design is evaluated against Triple Modular Redundancy (TMR)-based multiplication, unprotected precise multiplication, and unprotected approximate multiplication using hardware resource utilization and reliability metrics of the DNN accelerator. Experimental results demonstrate that, compared to the precise multiplier, the proposed architecture achieves 2.74× area reduction and a 39.06% lower power-delay product, while maintaining a reliability level close to TMR-protected multipliers. Furthermore, it provides effective fault detection and mitigation capabilities with precision comparable to state-of-the-art approximate multipliers in terms of area, delay, and power consumption.

Index Terms— Approximate Computing, Fault-Tolerant Multiplier, Deep Neural Networks, ASIC Accelerator, Leading One Detector (LOD), Verilog HDL, Xilinx Vivado.

I. INTRODUCTION

Deep Neural Networks (DNNs) have become fundamental in modern applications such as image processing, speech recognition, autonomous systems, and medical diagnosis. Despite their remarkable performance, DNNs demand high computational power while operating under strict energy constraints, particularly in edge and safety-critical environments.

The multiply-and-accumulate (MAC) operation forms the computational backbone of DNNs. As a result, multipliers constitute one of the most resource-intensive components in DNN hardware accelerators, significantly contributing to overall power consumption and silicon area. Although exact multipliers deliver high computational accuracy, they incur substantial hardware overhead in terms of power and area.

Fortunately, DNN applications exhibit inherent error resilience, allowing small computational inaccuracies without significantly degrading overall performance. This characteristic enables the adoption of approximate computing techniques to improve energy efficiency. Logarithmic multipliers based on the Mitchell algorithm are commonly employed in such designs due to their reduced hardware complexity and lower power consumption. However, conventional logarithmic multipliers suffer from notable approximation errors and lack adaptability and fault tolerance, limiting their suitability for safety-critical edge-AI systems.

To overcome these limitations, this work proposes FaTAM, a Fault-Tolerant Adaptive Approximate Logarithmic Multiplier designed for ASIC-based DNN accelerators. The proposed approach achieves an improved trade-off among accuracy, energy efficiency, and reliability, making it well-suited for

next-generation DNN accelerator applications.

I. EXISTING METHOD

Multiplication is a computationally intensive operation in digital systems, particularly in signal processing and neural network applications. To reduce hardware complexity, approximate techniques such as the Mitchell Logarithmic Multiplier (MLM) are widely used. In this method, multiplication is transformed into addition by representing input operands in logarithmic form, thereby simplifying the arithmetic operations involved.

In the MLM approach, each input operand is first processed using a Leading One Detector (LOD), which identifies the position of the most significant '1' bit. This position represents the exponent (characteristic), while the remaining lower-order bits form the mantissa (fractional part). An encoder converts the detected leading-one position into its corresponding binary exponent value. The mantissa is then normalized using a barrel left shifter, which aligns the fractional bits based on the detected exponent. Barrel shifters are preferred because they enable variable shifting within a single clock cycle, improving operational speed.

Once both operands are converted into logarithmic-like representations of the form:

$$X = 2^k(1 + f)$$

where k is the exponent and f is the fractional mantissa, the logarithmic addition stage begins. In this stage, the exponents of the two operands are added using a conventional adder, while the mantissas are also combined through addition. This effectively replaces multiplication with addition, significantly reducing hardware complexity.

The result is then passed to the antilogarithm unit, which converts the logarithmic result back into the binary domain. The final product is reconstructed using shift-and-add operations. Since logarithmic conversion introduces approximation, the output is an approximate product rather than an exact one.

A zero-detection block is included to check whether any input operand is zero. If a zero input is detected, the output is forced to zero, preventing invalid logarithmic operations and ensuring functional correctness.

The final output is generated as an approximate *n-bit* or *2n-bit* product, depending on the design configuration. Compared to exact multipliers, this approach achieves lower area and higher speed but at the cost of computational accuracy.

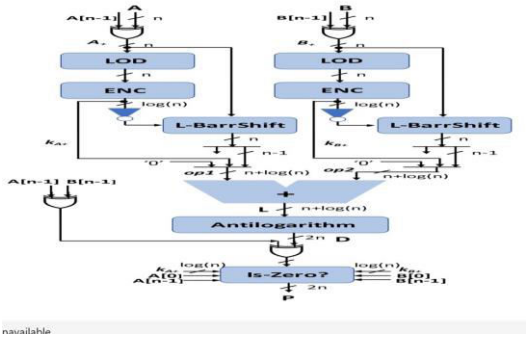

Fig.1 Block Diagram

1. **Input Operands** – Two *n-bit* binary inputs are provided simultaneously to the logarithmic processing blocks.
2. **Leading One Detector (LOD)** – Detects the position of the most significant '1' bit in each operand. This position represents the exponent.
3. **Encoder** – Converts the LOD output into a binary exponent value for logarithmic processing.
4. **Barrel Left Shifter** – Normalizes the mantissa by shifting it based on the detected leading-one position.
5. **Logarithmic Addition Unit** – Adds the exponents and mantissas using conventional adders, replacing multiplication with addition.
6. **Antilogarithm Unit** – Reconstructs the approximate product using shift-and-add operations.
7. **Zero Detection Block** – Ensures correct output by forcing the result to zero when any input is zero.
8. **Final Output** – Produces an approximate product with reduced hardware complexity and improved speed.

Drawbacks Of Existing Method

Despite its hardware efficiency, the Mitchell Logarithmic Multiplier suffers from several limitations:

- High approximation errors reduce computational accuracy for certain input combinations.
- Lack of adaptive error control prevents dynamic adjustment between accuracy and power consumption.
- No built-in fault-tolerant mechanisms,

making the system vulnerable to hardware faults and aging effects.

- Reduced long-term operational reliability due to absence of reliability-enhancement features.
- Not well-suited for modern DNN accelerators, where controlled approximation and reliability are essential.
- Error accumulation becomes significant in multi-stage operations such as MAC units in neural networks.
- No support for reliability–energy trade-offs, limiting flexibility in low-power and safety-critical applications.

IV PROPOSED METHOD

We propose an Adaptive Fault-Tolerant Approximate Multiplier (FaTAM) tailored for ASIC-based DNN accelerators. The architecture is derived from the classical Mitchell logarithmic multiplier; however, it introduces adaptive fault tolerance and enhanced approximation control. Although the design is built upon the Mitchell algorithm, the methodology can be extended to all logarithmic approximate multipliers.

The proposed architecture incorporates:

- An adaptive adder that utilizes an unconventional use of Leading One Detector (LOD) values for fault detection and mitigation.
- A gate-level optimized LOD design for reduced hardware overhead.
- A lightweight hybrid triplicated adder structure for enhanced reliability.

An additional level of approximation is introduced within the adaptive adder (AdAM), specifically optimized for DNN applications, where small computational inaccuracies have negligible impact on network accuracy.

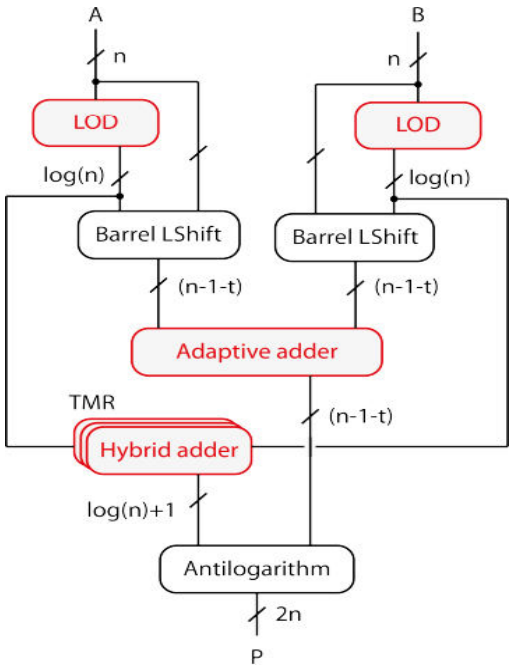

Fig.2 Block diagram of proposed system

A. Mathematical Foundation

For an n-bit operand A, let $k$denote the position of the leading one bit such that:

$$0 \leq k < n$$

By factoring out 2^k, the number can be expressed as:

$$A = 2^k(1 + X)$$

where $0 \leq X < 1$represents the fractional (mantissa) part.

The logarithm of $A$is:

$$\log(A) = k + \log(1 + X)$$

Mitchell's method approximates:

$$\log(1 + X) \approx X$$

Thus, the logarithm becomes:

$$\log(A) \approx k + X$$

For two operands $A$and B:

$$\log(A \cdot B) \approx (k_A + X_A) + (k_B + X_B)$$

The final product is obtained by taking the antilogarithm of the sum.

B. Adaptive Approximation (AdAM)

The AdAM architecture introduces a truncation parameter $t$that determines the minimum number of protected higher-order mantissa bits. The truncated fractional part is defined as:

X_t = Truncated fractional part based on t

Truncation introduces additional approximation error. However:

- No extra error occurs when truncated bits

are zero.

- Error increases as the number of '1's in truncated bits increases.

Since DNN synaptic weights are typically centered around zero, the practical impact of truncation error on overall network accuracy is minimal.

The fault tolerance of the design is controlled by two parameters:

- Truncation parameter (t): Minimum number of protected bits
- Duplication level (h): Maximum number of protected bits

Increasing $h$improves fault tolerance but increases area, power, and delay. Decreasing $h$reduces hardware cost. Different configurations are denoted as:

$$\mathrm{AdAM}(t, h)$$

C. Hardware Implementation

The FaTAM multiplier consists of three major blocks:

1. Optimized Leading One Detector (LOD)
2. Adaptive Fault-Tolerant Adder
3. Antilogarithm Unit

The LOD extracts exponent and mantissa components. The adaptive adder performs truncated mantissa addition with built-in fault detection and correction. A hybrid replicated adder with majority voting improves reliability during exponent addition. Finally, the antilogarithm unit reconstructs the approximate product using shift-and-add operations.

This hardware structure achieves:

- Reduced area
- Lower power consumption
- Improved fault tolerance
- High suitability for energy-efficient DNN accelerators

D. Adaptive Adder Architecture

The adaptive adder utilizes the exponent value $k$to dynamically adjust fault protection. It consists of three sections:

1. t Partial Full Adders (PFAs) – Duplicate addition of higher-order bits.
2. (h − t) PFAs with multiplexers – Adaptively protect higher or lower bits.
3. (n − 1 − h) PFAs – Perform normal mantissa addition.

For smaller operands:

- Fewer bits are required for computation.
- Unused adder resources are reallocated for fault protection.
- As $k$decreases, more bits are protected.

Example (8-bit inputs, $t = 2, h = 3$):

- When $k \leq 3$, all bits are protected.
- Provides full fault detection and mitigation.

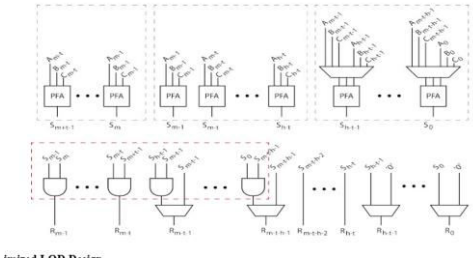

Fig.3 Architecture of proposed system

This adaptive reuse of hardware ensures efficient resource utilization while maintaining high reliability.

E. Optimized LOD Design

The proposed gate-level LOD consists of:

1. Leading-One Position Detection (LOPD)
2. Zero Flag Generation

The input is divided into nibbles and processed in parallel to detect the most significant '1' bit. For each nibble, control signals indicate:

- Presence of '1'
- Bit parity
- Higher-order bit activity

These signals are combined to identify the highest active nibble. The zero flag is derived from LOPD outputs, eliminating the need for separate zero-detection circuitry.

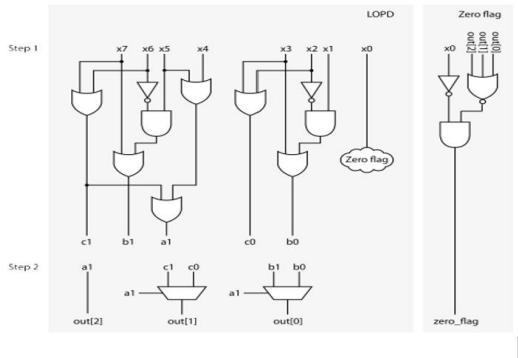

Fig.4 LOD of Proposed system

V RESULTS

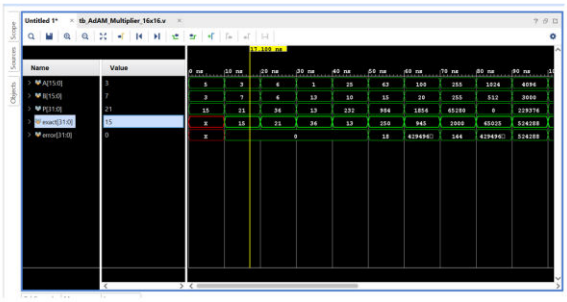

Fig.5 Simulation result

```
# )
# run 1000ns
================================================
 FINAL Hybrid AdAM 16x16 Multiplier Testbench
================================================
A=    5 B=    3 | Exact=        15 | Hybrid=        15 | Error=         0
A=    3 B=    7 | Exact=        21 | Hybrid=        21 | Error=         0
A=    6 B=    6 | Exact=        36 | Hybrid=        36 | Error=         0
A=    1 B=   13 | Exact=        13 | Hybrid=        13 | Error=         0
A=   25 B=   10 | Exact=       250 | Hybrid=       232 | Error=        18
A=   63 B=   15 | Exact=       945 | Hybrid=       984 | Error=       -39
A=  100 B=   20 | Exact=      2000 | Hybrid=      1856 | Error=       144
A=  255 B=  255 | Exact=     65025 | Hybrid=     65280 | Error=      -255
A= 1024 B=  512 | Exact=    524288 | Hybrid=         0 | Error=    524288
A= 4096 B= 3000 | Exact=  12288000 | Hybrid=    229376 | Error=  12058624
A=13604 B=24193 | Exact=  329121572 | Hybrid=     65536 | Error=  329056036
A=54793 B=22115 | Exact=1211747195 | Hybrid=    131072 | Error= 1211616123
A=31501 B=39309 | Exact=1238272809 | Hybrid=         0 | Error= 1238272809
A=33893 B=21010 | Exact= 712091930 | Hybrid=    131072 | Error=  711960858
A=58113 B=52493 | Exact=-1244441587 | Hybrid=    262144 | Error=-1244703731
A=61814 B=52541 | Exact=-1047197922 | Hybrid=         0 | Error=-1047197922
A=22509 B=63372 | Exact=1426440348 | Hybrid=         0 | Error=1426440348
A=59897 B= 9414 | Exact= 563870358 | Hybrid=    131072 | Error= 563739286
A=33989 B=53930 | Exact=1833026770 | Hybrid=    393216 | Error=1832633554
A=63461 B=29303 | Exact=1859597683 | Hybrid=    262144 | Error=1859335539
```

Fig.6 Simulation result

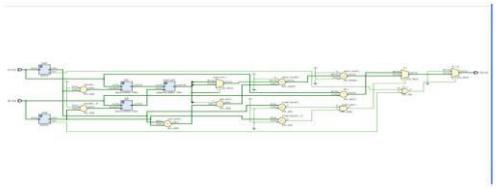

Fig.6 Elaboration Design

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	169	41000	0.41
DSP	1	240	0.42
IO	64	300	21.33

Fig.7 Area

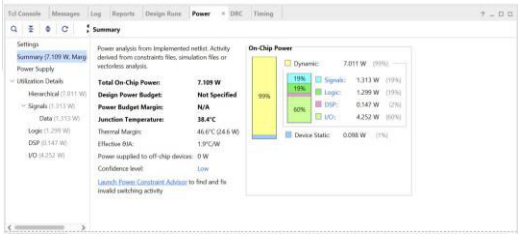

Fig.9 Power

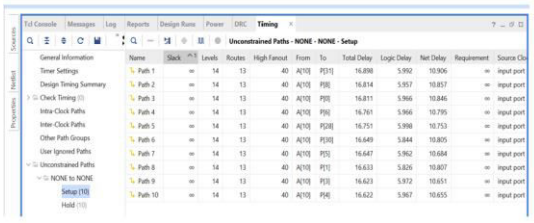

Fig.10 Delay

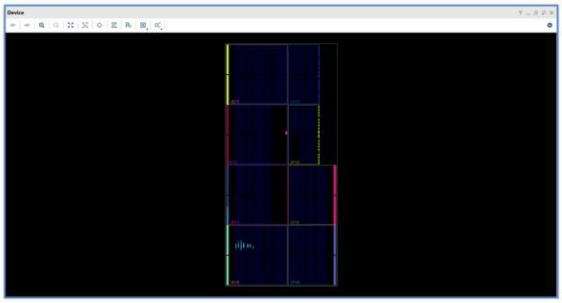

Fig.11 Implemented Design

Compression Table

Parameter	Existing Method (Conventional LM)	Proposed FaTAM
Area (LUT)	245	169
Power(W)	11.8	7.01
Delay(ns)	21	16

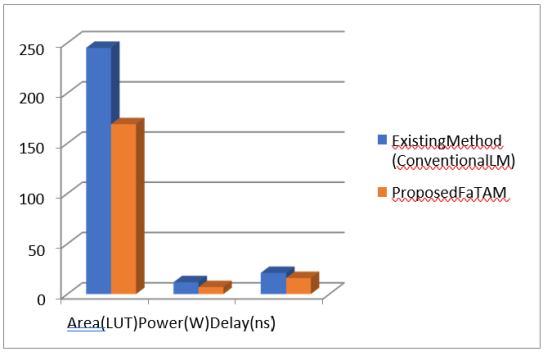

Fig.12 Comparison analysis

VI RESULT ANALYSIS

From the comparative analysis, it is observed that the proposed FaTAM architecture significantly outperforms the conventional logarithmic multiplier in multiple performance metrics. The substantial area reduction is achieved through the use of optimized

arithmetic units and simplified computation logic, which minimize redundant hardware resources.

Power consumption is considerably reduced by replacing complex exact multiplication circuitry with adaptive approximate operations, thereby lowering switching activity and overall hardware complexity. The observed delay reduction further indicates improved operational speed, making the proposed architecture well-suited for high-throughput DNN accelerator applications.

Moreover, unlike the existing logarithmic multiplier, the proposed FaTAM incorporates built-in fault-tolerance mechanisms. This enhancement significantly improves system reliability and operational robustness, which are critical requirements for modern safety-critical and energy-efficient DNN accelerators.

VII CONCLUSION

In this work, FaTAM: A Fault-Tolerant Adaptive Approximate Logarithmic Multiplier for Energy-Efficient DNN Accelerators has been presented to address the increasing demand for low-power and high-performance arithmetic units in modern neural network hardware. The proposed architecture builds upon the conventional logarithmic multiplier by incorporating adaptive approximation and fault-tolerant mechanisms, thereby overcoming the key limitations of existing designs.

By transforming multiplication into simpler addition and shift operations, the proposed multiplier significantly reduces hardware complexity, area, and power consumption. The introduction of adaptive adder structures enables dynamic control of approximation error based on input characteristics, ensuring improved computational efficiency without severely impacting accuracy. Furthermore, the integration of lightweight fault-tolerance techniques enhances system reliability under hardware faults, aging effects, and process variations.

As a result, FaTAM achieves a superior balance among energy efficiency, computational accuracy, and robustness when compared to traditional exact multipliers and conventional approximate multipliers. The proposed design is particularly well-suited for DNN accelerators, where large numbers of multiply-and-accumulate (MAC) operations are performed and minor computational inaccuracies are inherently tolerable.

Overall, FaTAM demonstrates that the integration of approximate computing, adaptivity, and fault tolerance provides an effective and practical solution for designing energy-efficient and reliable arithmetic units for next-generation AI hardware systems.

REFERENCES

[1] J. N. Mitchell, “Computer multiplication and division using binary logarithms,” *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, Aug. 1962.

[2] E. E. Swartzlander, Jr., *Computer Arithmetic: Volume I*, IEEE Computer Society Press, 2000.

[3] S. Hashemi, R. I. Bahar, and S. Reda, “DRUM: A dynamic range unbiased multiplier for approximate applications,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 418–425.

[4] H. Jiang, J. Han, and F. Lombardi, “A comparative review and evaluation of approximate adders,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 4, 2017.

[5] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[6] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, “Design and analysis of approximate compressors for multiplication,” *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, Apr. 2015.

[7] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, “Approximate computing for energy-efficient deep learning systems,” *IEEE Design & Test*, vol. 34, no. 6, pp. 8–17, Dec. 2017.

[8] S. Rehman, M. Shafique, F. Kriebel, and J. Henkel, “Reliable and energy-efficient architectures for error-resilient computing,” *Journal of Low Power Electronics and Applications*, 2014.