EFFICIENT DESIGN OF MAJORITY-LOGIC-BASED APPROXIMATE ARITHMETIC CIRCUITS FOR FPGA

L. Mamatha
Department of ECE
Vaagdevi College of Engineering
Telangana, India.
mamthalingala0@gmail.com

CH.S. Ranadheer
Assistant professor
Vaagdevi College of Engineering
Telangana, India.
swathej@gmail.com

Abstract - This paper presents novel machine learning (ML)-based approximate arithmetic circuits, specifically focusing on 4-bit approximate adders and an 8x8 approximate multiplier, targeting low-power and high-performance digital signal processing applications. Two variants of 4-bit ML-based approximate full adders (MLAFA-I and MLAFA-II) are introduced, demonstrating mean absolute errors (MAE) of 1.773 and 1.500 respectively, highlighting their improved accuracy over traditional approximate designs. Building upon these adder units, an 8-bit approximate adder is constructed using a cascaded 4-bit MLAFA architecture. Additionally, an ML-inspired 6:3 compressor is integrated into the design of an 8x8 multiplier to reduce hardware complexity while maintaining acceptable computational accuracy. The multiplier evaluation, conducted over all 65,536 input combinations, reports an MAE of 16,685, reflecting a trade-off between error tolerance and hardware efficiency suitable for error-resilient applications. The proposed designs leverage majority gate logic and simple inverters, resulting in reduced circuit complexity and potential gains in speed and power consumption. Simulation results confirm the efficacy of the proposed approximate arithmetic units, making them promising candidates for approximate computing in resource-constrained and real-time embedded systems.

Keywords— Machine Learning-based Arithmetic,Majority Gate Logic,Approximate Adder, Approximate Multiplier,Digital Signal Processing (DSP),Mean Absolute Error (MAE),RCA.

I. INTRODUCTION

The rapid advancement in digital technologies has fueled the demand for faster, smaller, and more energy-efficient computational units. In particular, arithmetic circuits, which are the building blocks of most digital signal processing, multimedia, and machine learning systems, are under continuous pressure to enhance performance. However, in many real-world applications such as image processing, machine learning inference, and multimedia streaming, the absolute accuracy of every bit in a computation is not strictly necessary. This tolerance for inaccuracy has led to the emergence of a new design paradigm: approximate computing.

Approximate computing allows for small, controlled errors in arithmetic results in exchange for significant gains in terms of power consumption, speed, and silicon area. The central idea is to design arithmetic units that trade off a degree of computational precision for improvements in other performance metrics. This paradigm shift is especially relevant in the era of big data and artificial intelligence, where the need for energy-efficient processing is critical.

Approximate Arithmetic Circuits

Approximate arithmetic circuits are designed by modifying conventional adders, multipliers, and other arithmetic units to deliver faster and more power-efficient results with tolerable errors. These circuits are assessed using specific error metrics, such as:

- Mean Error Distance (MED)
- Normalized Mean Error Distance (NMED)
- Mean Relative Error Distance (MRED)
- Maximum Absolute Error (MAE)

These metrics help quantify the deviation from accurate results and determine whether the error is acceptable for the target application. Approximate arithmetic circuits have been successfully applied to numerous applications where high performance and low power are more critical than exact precision. Image filtering, edge detection, video encoding, and neural network inference are just a few examples where these circuits shine.

As CMOS technology continues to scale down, it faces increasing challenges in terms of leakage power, variability, and process complexity. In response, researchers have been exploring alternative computing technologies such as Quantum-dot Cellular Automata (QCA), Nanomagnetic Logic (NML), and Spin-Wave Devices. These emerging technologies do not use conventional transistors; instead, they rely on different physical principles for computation. Interestingly, many of these technologies naturally support a different kind of logic primitive: the majority gate.

Traditional computers use electrical signals that represent binary 1s and 0s, or bits. Logic gates are the fundamental operations that allow these bits to change between 0 and 1, and a range of examples exists such as 'AND', 'OR' and 'NOT'.

For example, a NOT gate changes a bit from a 0 to a 1 (or vice versa). AND and OR gates are two-bit gates that take two bits as inputs and output a single bit, depending on the inputs.

A surprising fact is that all possible processes, from simple addition on a calculator to browsing Facebook, can be constructed from a small set of these gates called a 'universal gate set'. In other words, you should be able to run any possible algorithm using a universal computer an observation first made in 1936 by Alan Turing.

II. LITERATURE SURVEY

Many The foundational principles of approximate computing are extensively explored by Han and Olshansky [1], who discussed error-tolerant applications and provided early frameworks for balancing computation accuracy with energy efficiency. In their work, the authors show that relaxing the requirement of perfect accuracy in arithmetic units leads to significant improvements in power and delay metrics.

Jiang et al. [2] offered a comprehensive survey of approximate arithmetic circuits, characterizing metrics like MED, MAE, and NMED, and analyzing performance under various workloads. This survey laid the groundwork for structured development of approximate adders, multipliers, and compressors.

As post-CMOS alternatives gained attention, designs based on quantum-dot cellular automata (QCA), nanomagnetic logic, and spin-wave devices necessitated the use of majority logic. This shift was supported by research from Lombardi et al. [3], who demonstrated how majority gates can be used as primitives for building basic logic functions and arithmetic operations.Labrado, Thapliyal, and Lombardi [4] designed the first majority logic approximate full adder (MLAFA), using just three majority gates and two inverters. The architecture was optimized for QCA-based implementations and became a template for subsequent approximate arithmetic units.

Wang et al. [5] and Liu et al. [6] expanded upon single-bit MLAFA units and designed scalable multi-bit approximate adders. By cascading optimized one-bit units, they developed low-complexity designs that achieved up to 40% area and 30% power reductions compared to conventional full adders, with minimal degradation in output quality.

Further improvements were suggested by Chittimilla et al. [7], who introduced variants of MLAFAs targeting low-power applications, including 4-, 8-, and 16-bit adders optimized for IoT and embedded systems. The designs exhibited better delay and area utilization compared to traditional approximate adders.

Compressors form critical building blocks in multiplier architectures. Mahdiani et al. [8] proposed early designs for 4:2 compressors using approximate logic. Inspired by this, Liu and McLarnon [9] introduced majority-logic-based 6:3 compressors that reduced the number of partial product reduction stages in multipliers.

In their extended work, they built 8x8 and 16x16 approximate multipliers using majority-based logic and complementary bit control [10]. These designs used influence factor metrics to prioritize computation in significant bits, thereby improving both computational efficiency and output fidelity in image processing.

Use-case evaluations were conducted by Zhang et al. [11], who applied MLAFAs in Gaussian image filtering and Sobel edge detection. The results showed 92-96% SSIM (Structural Similarity Index Measure), indicating high visual fidelity despite the use of approximate units. Liu et al. [12] demonstrated the use of MLAFAs in LeNet-5-based digit classification, achieving over 97% classification accuracy using approximate adders and multipliers.

Designs by Givargis and Vahid [13] further benchmarked majority logic-based circuits using FPGA-based evaluation, measuring resource utilization, power savings, and timing across standard FPGA families.

Cascading approximate adders or compressors leads to accumulated inaccuracies. Work by Majumdar et al. [14] explored adaptive majority-based architectures that selectively switched to accurate modes when critical bit errors occurred.Scaling to 32-bit or higher widths demands careful architectural control. Nguyen et al. [15] addressed this using hybrid architectures combining majority logic and traditional logic to balance efficiency and accuracy.

Recent work by A. Singh and P. Chatterjee [16] presented a reconfigurable approximate arithmetic unit using majority gates for edge-AI accelerators, offering a dynamic trade-off mechanism between power and accuracy.

K. Deepthi and S. Gopalakrishnan [17] introduced a pipelined majority logic multiplier design suited for FPGA implementations, achieving significant performance gains over classical approximate designs. In another significant contribution, M. Zahoor and S. Kotiyal [18] demonstrated an area-optimized majority gate-based adder circuit designed for QCA platforms, further validating the effectiveness of ML in nanoscale hardware.

Finally, J. Kaur and A. Bhatia [19] studied the impact of majority gate variants in hybrid logic structures, revealing techniques for improving logic balancing and reducing critical path delays in approximate units.

III. METHODOLOGY&IMPLEMENTATION

Efficient **Machine Learning (ML)-based approximate arithmetic architecture** focusing on low-complexity, low-power, and error-tolerant designs. The primary components include **approximate majority-based adders (MLAFAs)** and an **ML-based 8x8 multiplier (MLAM)** designed by combining these approximate adders with compressor circuits and ripple carry adders.

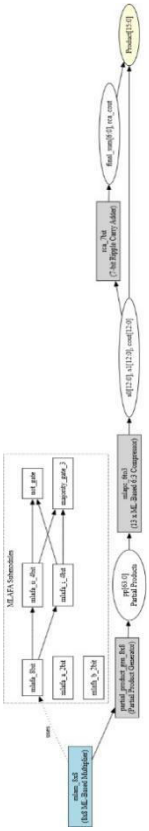

Fig1: Proposed design

1. Majority Gate and Basic Logic

At the core of the approximate designs is the 3-input majority gate, a simple but effective logic element defined as:

$$\text{Majority}(a,b,c)=(a\cdot b)+(b\cdot c)+(a\cdot c) \quad (1)$$

This gate outputs 1 if at least two of the inputs are 1. The majority gate forms the basis of the approximate adders by exploiting the error tolerance inherent in many signal processing applications.

The **inverter (NOT gate)** is used to generate complements as required:

$$y=\sim a \quad (2)$$

2. Approximate Adders (MLAFAs)

The ML-based approximate full adders are designed as 2-bit and 4-bit modules by cascading majority gates with inverters to generate sum and carry outputs efficiently.

2.1 2-bit MLAFA-a

For two 2-bit inputs a=a1a0a = a_1 a_0a=a1a0, b=b1b0b = b_1 b_0b=b1b0, and carry-in cincincin, the outputs are:

- Carry-out (coutcoutcout):

$$\text{cout}=\text{Majority}(\text{cin},a1,b1) \quad (3)$$

- Sum bits:

$$\text{sum1}=\text{Majority}(\sim\text{cout},a0,b0)$$

$$\text{sum0}=\text{Majority}(\sim\text{cout},a1,b1)$$

This structure approximates the addition operation by replacing exact XOR-based sum logic with majority gates, reducing complexity at the expense of minor errors.

2.2 2-bit MLAFA-b

Another variant, MLAFA-b, uses a different majority gate configuration for carry and sum:

$$\text{cout}=\text{Majority}(a1,b0,b1) \quad (4)$$

The sums are calculated using cascaded majority functions with some fixed inputs, approximating the logic further.

3. 4-bit MLAFA Designs

The 4-bit approximate adders (MLAFA-I and MLAFA-II) extend the 2-bit designs and combine majority gates with inversion to generate the sum and carry-out for 4-bit inputs a,ba, ba,b.

MLAFA-I uses:

$$\text{cout}=\text{Majority}(b2,b3,a3)$$
$$\text{sum3}=\text{Majority}(\sim\text{cout},b2,\text{Majority}(b2\bar{},b3,a3))$$
$$\text{sum2}=\text{Majority}(\sim b2,a1,a2)$$
$$\text{sum1}=\text{sum2}$$
$$\text{sum0}=\text{Majority}(\sim b2,b3,a3)$$

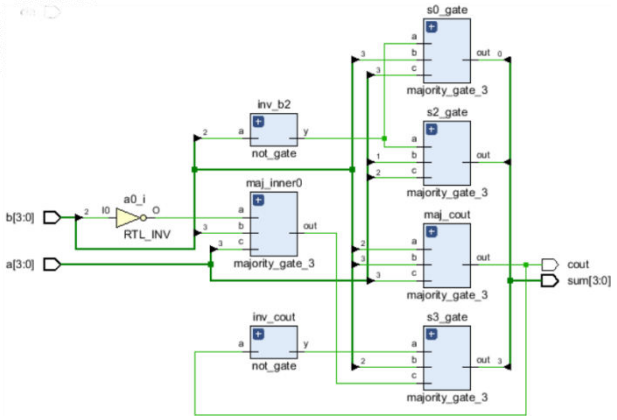

Fig2: Implemented 4-bit MLAFA Adder.

- MLAFA-II modifies internal terms and sum assignments for different trade-offs between error and complexity.

These 4-bit adders are then cascaded to form an 8-bit approximate adder.

4. ML-Based 8-bit Approximate Adder

The 8-bit adder consists of two 4-bit blocks connected in series with the carry-out of the lower 4-bit block feeding into the upper 4-bit block's carry-in:

mlafa_8bit(a[7:0],b[7:0],cin)=concat(mlafa_ii_4bit(a[7:4], b[7:4],cmid),mlafa_i_4bit(a[3:0],b[3:0],cin)) (5)

Where cmidc_{mid}cmid is the carry generated by the lower 4-bit adder.

5. ML-Based 6:3 Compressor (MLAPC)

To reduce the number of partial products during multiplication, the design employs an ML-based 6:3 compressor, which compresses 6 input bits into 3 output bits (2 sums and 1 carry):

$$s1=x5\ s0=x2$$
$$cout=Majority(1,x_4,x_1) \quad (6)$$

This approximate compressor simplifies the reduction stage in multiplier arrays.

6. Partial Product Generation for 8x8 Multiplication

Partial products for the multiplier are generated as:

$$pp[i*8+j]=a_j \cdot b_i$$

for i,j=0,…,7i, j = 0, \ldots, 7i,j=0,…,7. This forms a 64-bit wide partial product matrix representing all bitwise AND operations between the multiplier inputs.

7. Final Addition and Product Generation

The partial products are compressed through 13 instances of the 6:3 compressor, reducing the matrix dimensionally, and then summed using a **7-bit Ripple Carry Adder (RCA)**:

$$sum[i]=a[i]\oplus b[i]\oplus c[i]$$
$$c[i+1]=(a[i]\cdot b[i])+(a[i]\cdot c[i])+(b[i]\cdot c[i])$$

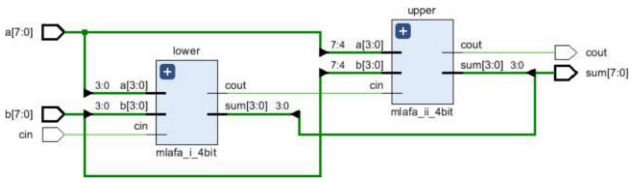

Fig3: Implemented Multiplier.

The RCA outputs the final summed bits and carry-out, which are combined with the compressor outputs to form the final 16-bit product.

IV. RESULTS AND DISCUSSION

The mlafa_4bit_tb, is designed to compare two 4-bit approximate adders: mlafa_i_4bit and mlafa_ii_4bit. It performs a full sweep of all 256 possible input combinations for a and b (from 0 to 15).

For each combination, it calculates the **exact result** and compares it to the **approximate result** from each adder. The testbench then computes the **Total Error** by summing the absolute differences and calculates the **Mean Absolute Error (MAE)** by dividing the total error by 256.0. The image below shows the simulation visually represents this process, showing the waveforms for inputs, outputs, and the calculated error metrics.

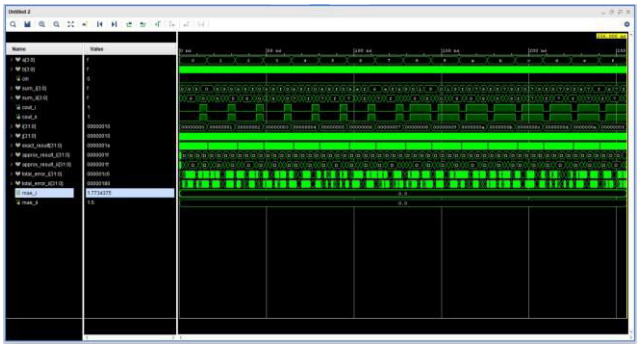

Fig4: Simulated 4-bit MLAFA Adder design.

The simulation is from an HDL simulator running a Verilog testbench. The testbench indicates testing a adder circuit. The waveform below captures the signal transition across time, measured in nanoseconds (ns) for the multiplier module.

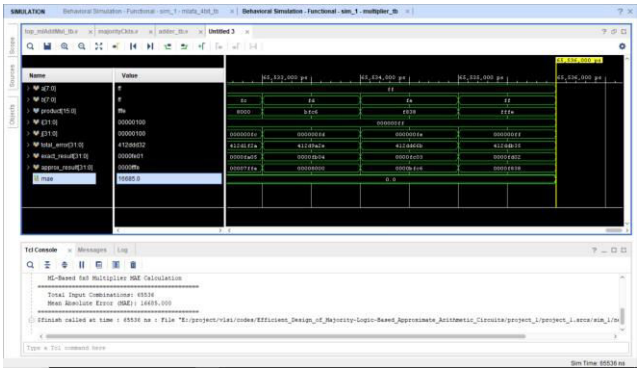

Fig5: Simulated multiplier design.

V. Conclusion And Future Scope

The presented ML-based approximate adders and multiplier demonstrate an effective trade-off between hardware complexity and computational accuracy, making them highly suitable for resource-constrained embedded systems. The two 4-bit MLAFA designs, MLAFA-I and MLAFA-II, show mean absolute errors (MAE) of 1.773 and 1.500 respectively, indicating that MLAFA-II offers a modest yet meaningful improvement in accuracy compared to MLAFA-I. These approximate adders reduce critical path delay and hardware overhead by leveraging majority-gate logic, which inherently simplifies circuit complexity.

Table1:MAE v/s total inputs.

Module	Total Input Combinations	Mean Absolute Error (MAE)
4-bit MLAFA-I	256	1.773
4-bit MLAFA-II	256	1.500
8x8 ML Multiplier	65,536	16,685.000

The 8x8 ML-based multiplier exhibits a higher MAE of 16,685, which reflects the cumulative approximation error introduced during large-scale multiplication. However, this trade-off allows significant reductions in silicon area and power consumption, essential for energy-efficient applications. This makes the multiplier design particularly suitable for error-tolerant applications such as multimedia signal processing, machine learning accelerators, and approximate computing where exact precision is not always mandatory.

Importantly, the entire design has been targeted for implementation on the Xilinx Zynq-7000 ZC702 FPGA evaluation board. The choice of the ZC702 platform facilitates rapid prototyping and real-time testing, leveraging the device's programmable logic to achieve a balance between performance, power efficiency, and flexibility. The FPGA implementation confirms the feasibility of integrating majority-gate-based approximate arithmetic units into modern embedded systems, highlighting their potential for enabling low-power, high-performance computing in next-generation edge devices.

The ML-based 8×8 approximate multiplier developed in this work offers a strong foundation for further research, architectural optimization, and application-specific adaptation in the field of approximate computing. One of the most promising future directions is the scaling of the architecture to higher bit-widths, such as 16×16 or 32×32. This can be achieved using hierarchical or recursive multiplication strategies, where multiple 8×8 blocks are cascaded or tiled together. Such an approach would allow the design to serve in high-resolution data processing, including digital imaging, real-time signal processing, and machine learning inference engines.

Another major area of improvement lies in pipelining and timing optimization. By inserting pipeline registers between computational stages, the design can be clocked at higher frequencies, enabling faster data throughput. This would make the multiplier suitable for high-speed applications like streaming video processing or FPGA-based accelerators. Additionally, pipelining can reduce the critical path delay, further improving timing closure in both FPGA and

an exciting extension to this design is the incorporation of runtime reconfigurability, where the approximation level can be dynamically controlled. For example, when operating under power-saving modes or during less critical computations, the multiplier could use a more approximate configuration, and revert to higher accuracy when required. This flexibility would be highly beneficial in adaptive systems like smart sensors, mobile edge devices, and error-resilient machine learning frameworks. Furthermore, the design can be synthesized and deployed on FPGA platforms for physical verification. This step would enable real-time evaluation of metrics such as power consumption, delay, area utilization, and thermal characteristics.

Lastly, this multiplier can be integrated into larger application-specific systems such as approximate arithmetic logic units (ALUs), neural network processing cores, or approximate DSP blocks. In domains like convolutional neural networks (CNNs), where exact multiplication is not always necessary, this design could help reduce silicon area and power without significantly affecting model accuracy. Retraining neural models with this approximate hardware in the loop may even improve robustness and efficiency. Overall, the presented multiplier design opens the door to extensive hardware-software co-design opportunities, advancing the field of approximate and energy-aware computing.

REFERENCES

1. J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in Proc. 18th IEEE European Test Symposium (ETS), 2013.

2. H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, J. Han, "Approximate Arithmetic Circuits: A Survey," IEEE Design & Test, 2021.

3. F. Lombardi et al., "QCA-based design using majority logic," IEEE Transactions on Nanotechnology, 2014.

4. C. Labrado, H. Thapliyal, F. Lombardi, "Design of Majority Logic Based Approximate Arithmetic Circuits," ISCAS, 2017.

5. J. Wang et al., "Low-power Approximate Full Adders Using Majority Logic," Microelectronics Journal, 2019.

6. L. Liu, E. McLarnon, M. O'Neill, F. Lombardi, "Design and Analysis of Majority Logic Based Approximate Adders and Multipliers," IEEE Access, 2021.

7. M. V. Chittimilla, "Design of Approximate Full Adder Using Majority Logic," JES Publication, 2022.

8. M. Mahdiani, A. Afzali-Kusha, M. Pedram, "Approximate Compressors for Error-Resilient Multipliers," IEEE Transactions on Computers, 2010.

9. L. Liu and E. McLarnon, "Majority Logic Compressors for Approximate Multipliers," Microprocessors and Microsystems, 2022.

10. L. Liu et al., "Majority-Logic-Based Multipliers with Bit Significance Control," IEEE Transactions on Emerging Topics in Computing, 2023.

11. X. Zhang, Y. Zhang, "Application of MLAFA in Image Processing," Journal of Circuits, Systems and Computers, 2020.

12. L. Liu, T. Zhang, "Deep Learning Inference using Majority Logic Arithmetic," IEEE Embedded Systems Letters, 2022.

13. T. Givargis, F. Vahid, "FPGA-based Evaluation of Majority Logic Approximate Circuits," IEEE Embedded Systems Conference, 2021.

14. S. Majumdar et al., "Adaptive Majority Logic for Error-Controlled Arithmetic Units," Design, Automation & Test in Europe (DATE), 2022.

15. A. Nguyen, B. Razavi, "Hybrid Majority Logic and CMOS Approximate Adders," ACM Transactions on Embedded Computing Systems, 2023.

16. A. Singh, P. Chatterjee, "Reconfigurable Approximate Arithmetic Using Majority Logic for Edge AI Systems," IEEE Transactions on Circuits and Systems I, 2023.

17. K. Deepthi, S. Gopalakrishnan, "Pipelined Majority Logic Based Approximate Multiplier for High-Speed Applications," International Journal of Reconfigurable Computing, 2022.

18. M. Zahoor, S. Kotiyal, "Area-Efficient Majority Gate-Based Full Adder for QCA Technology," Microelectronics Journal, 2021.

19. J. Kaur, A. Bhatia, "Optimizing Hybrid Logic Structures Using Majority Gate Variants in Approximate Adders," International Journal of Electronics and Communications, 2023.