1 Abstract

The energy-efficient computing-in-memory (CIM) architectures have drawn much attention as the increasing demands of neural networks. Several SRAM-based CIM architectures adopt a digital implementation, using the digital adder trees (ATs) to perform in-memory multiply-accumulate (MAC) operations. Compared with the analog-domain CIM, the digital CIM eliminates errors caused by analog circuits to achieve high accuracy. However, the digital AT still incurs much power and area overhead. This brief proposes a novel low-power AT solution by approximate circuit co-design. Two approximate full adder logic is substituted to the full-adders in the wallace tree structure to enhance the performance regarding consuming low power and low area . The first proposed approximate wallace tree multiplier scheme achieves 33.05% reduction in area and a 25.9% where another one achieves 25.75% reduction in power and 34.57% reduction in area. Approximate adders effectiveness is determined in computing DCT(Discrete Continuous Transform) of benchmark image lena where the achieved PSNR of the approximate adder is 49.81 dB that allows slide error from the exact adder but the SSIM is almost same which serves the purpose of DCT.

2 Introduction

Computational In-Memory(CIM) is a computing architecture that performance computations directly within memory rather than relying on traditional Von Neumann architecture, where data must be transfer between memory and the processor. CIM aims to reduce data transfer latency and power consumption. Here adder tree plays a crucial role in accelerating arithmetic operation directly within memory. A wallace tree is widely used in Computation In Memory architectures especially in multiply-accumulate(MAC) operations for AI and Signal Processing application. It allows parallelism, meaning that multiple additions are performed simultaneously.

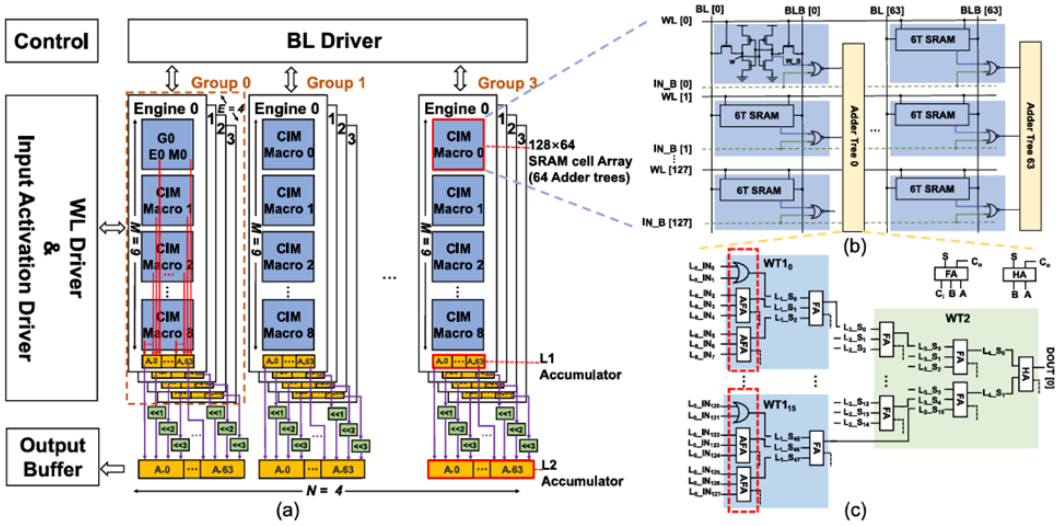


Figure 1: CIM Architecture

In the realm of modern computing, the relentless pursuit of energy efficiency and high performance has catalyzed the development of approximate computing techniques, particularly

in applications where minor inaccuracies are acceptable. This paradigm shift is especially pertinent in error-tolerant domains such as multimedia processing, machine learning, and scientific computing, where approximate computing leverages inherent error resilience to achieve substantial reductions in power consumption and silicon area, thereby enhancing overall computational efficiency.

Multiplication is a cornerstone operation in numerous computing tasks, and the Wallace tree multiplier stands out for its high-speed arithmetic capabilities. Introduced by Chris Wallace in 1964, the Wallace tree multiplier employs a hardware-efficient method to multiply two integers by summing partial products in stages using a selection of full and half adders. This approach reduces the number of sequential adding stages, thereby accelerating the multiplication process compared to naive methods. The Wallace tree algorithm operates in three primary steps:

Generation of Partial Products: Each bit of one operand is multiplied by each bit of the other operand, resulting in a matrix of partial products.

Reduction of Partial Products: The partial products are reduced to two rows by employing layers of full and half adders, known as the Wallace reduction. This step significantly decreases the number of addition stages required.

Final Addition: The two remaining rows are summed using a conventional adder to produce the final product.

The efficiency of the Wallace tree multiplier lies in its ability to perform reductions in parallel, leading to a logarithmic reduction in the number of partial product rows, which contributes to its high-speed performance.

While the traditional Wallace tree multiplier offers speed advantages, its conventional design can be resource-intensive, leading to increased power consumption and larger area requirements. To address these challenges, researchers have explored approximate versions of the Wallace tree multiplier that incorporate imprecise arithmetic units during the partial product reduction phase. This approach effectively balances the trade-off between computational accuracy and resource efficiency.

For instance, a study introduced an Approximate Reduced Complexity Wallace Multiplier (ARCWM) by replacing standard full adders with approximate ones. This modification resulted in notable improvements in area utilization, delay, and power consumption, demonstrating the potential of approximate multipliers in energy-efficient computing applications.

Integration into Computing-In-Memory (CIM) Architectures

The integration of approximate Wallace tree multipliers within computing-in-memory (CIM) architectures presents a promising avenue for enhancing computational efficiency. CIM architectures aim to mitigate the von Neumann bottleneck by performing computations directly within memory arrays, thereby reducing data movement and associated energy costs. Incorporating approximate multipliers into CIM systems can further optimize power and area efficiency, making them highly suitable for large-scale data processing tasks inherent in artificial intelligence and machine learning applications.

Designing approximate Wallace tree multipliers necessitates careful consideration of the trade-offs between accuracy, power consumption, and area efficiency. One effective methodology involves the use of genetic algorithms to explore the vast search space created by different combinations and placements of approximate adders. This approach allows for the optimization of multipliers tailored to specific application requirements. MDPI.COM

Another innovative design incorporates a Square Root Carry Select Adder (SQRT CSLA) with a mirror adder into the Wallace tree multiplier architecture. This configuration achieves high performance and reduced area by optimizing the adder block within the multiplier design. The proposed design demonstrated reduced delay and higher performance compared to

conventional multipliers using carry-save adders with majority-based gate adder logic.

Approximate Wallace tree multipliers are particularly advantageous in error-tolerant applications where slight inaccuracies are permissible in exchange for significant gains in power and area efficiency. For example, in image processing applications, approximate multipliers have been shown to reduce power consumption and critical path delay while maintaining acceptable levels of image quality. One study reported that an 8-bit approximate multiplier reduced power consumption and critical path delay by 36.9% and 38.9%, respectively, with a normalized mean error distance of 0.25%. The silicon area required to implement the multiplier was also reduced by 50.3%

The International Technology Roadmap for Semiconductors (ITRS) [1] has anticipated imprecise/approximate designs that became a state-of-the art demand for the emerging class of killer applications that manifest inherent error-resilience such as multimedia, graphics, and wireless communications. In the error-resilience systems, adders and multipliers are used as basic building blocks and their approximate designs have attracted significant research interest recently. Conventional wisdom investigated several mechanisms such as truncation [2], over-clocking, and voltage over-scaling(VOS) [3] which could not configure accuracy as well as Latency-Power-Area (LPA) design metrics effectively. Most of the other design techniques rely on functional approximations and a wide spectrum of approximate adders like [4], [5], [6] and [7] have been proposed in the past. However, very few research papers are reported on approximate multipliers in the literature. Most of the approximate multiplier designs reported shorten the carry-chains in which error is configurable and the algo rithms employed in the designs are for smaller numbers and give large magnitude of error as the bit-width of operands increases. In this paper, we present a new Approximate Wallace Tree Multiplier (AWTM) based on a bit-width aware algorithm. We design it specifically to give good results for large operands. Besides accuracy, the AWTM is also optimized for power and area. For single cycle implementation, AWTM gives significant reduction in latency as well.

The comprehensive analysis in[1] shows that the chosen approximate adder for ATWM are pareto optimal design choice for multiplier application. We analysis the electrical and error performance of 4*4 wallace tree multiplier by using the chosen two approximate adder.

3 Literature Review

Approximate computing is a design paradigm that relaxes accuracy constraints to achieve gains in power efficiency, area reduction, and speed. This concept is particularly useful in error-resilient applications such as image processing, artificial intelligence, and multimedia processing. Approximate arithmetic circuits, such as Approximate Full-Adders (approxFAs), play a crucial role in designing low-power computing architectures [1].

Since the full adder (FA) is a fundamental component in arithmetic circuits, including adders and multipliers, researchers have extensively explored various approxFA designs to balance trade-offs between accuracy and hardware efficiency. Approximate Wallace Tree Multipliers leverage these approxFAs to improve power, area, and delay performance while maintaining an acceptable level of computational accuracy [**?**]

Ettore Napoli et al. conducted an exhaustive study on approxFAs by modifying the truth table of a binary full adder, exploring a vast design space of 13,055 different circuits. Their key findings include:

- Pareto Optimal ApproxFAs: Only a limited subset of approxFAs are Pareto optimal in terms of power-error and area-error trade-offs. - 0-in/0-out Property: The research emphasized

the importance of the 0-in/0-out property in approxFAs. If an approximate FA does not return a zero output for an all-zero input (0,0,0), it can lead to incorrect results in applications like binary multipliers. - Error-Resilient Applications: The identified Pareto-optimal approxFAs demonstrated improved performance in practical applications such as image processing and neural network computations.

The study systematically extracted the best-performing approxFAs on the Pareto front by optimizing area vs. error and power vs. error trade-offs. The authors analyzed:

- Sum of Absolute Errors (SAE) and Mean Relative Error Distance (MRED) as key error metrics. - Silicon area occupation, power dissipation, and maximum delay as electrical performance metrics.

The results indicate that many previously proposed approximate full adders are suboptimal when implemented in a practical FinFET 14 nm standard cell library. The findings highlight that only a small subset of approxFAs offer an optimal trade-off between hardware and computational efficiency

A critical requirement for certain applications, including approximate multipliers, is that an FA must output (0,0) when given an input of (0,0,0). The study investigated 3,263 approxFAs that satisfy this requirement and extracted their corresponding Pareto fronts.

Without this property, binary multipliers and adders could generate nonzero outputs for trivial operations, leading to computational inaccuracies in hardware designs. The study confirms that the subset of approxFAs with this property provides more reliable results while maintaining the benefits of approximation.

The effectiveness of the identified Pareto-optimal approxFAs was validated in three key applications: - Neural Networks: ApproxFAs were used in approximate multipliers within a neural network inference model. Results showed that the best approxFAs reduced power consumption by up to 38% while maintaining over 90% classification accuracy. - Image Addition: ApproxFAs were integrated into Ripple Carry Adders for image blending applications, achieving power savings of up to 47% with minimal degradation in image quality. - Image Blurring: Approximate arithmetic circuits were employed in Gaussian filtering applications, reducing power consumption by up to 82% while preserving high structural similarity (SSIM) and peak signal-to-noise ratio (PSNR).

The study by Napoli et al. provides an extensive analysis of the design space of Approximate Full-Adders, identifying Pareto-optimal designs suitable for Approximate Wallace Tree Multipliers and other arithmetic circuits. By considering both error performance and hardware efficiency, the study offers a precise set of optimal approxFAs, eliminating the need to explore the entire design space.

The findings reinforce the importance of Pareto optimization and the 0-in/0-out property in selecting the best approxFAs for approximate computing architectures. These insights can guide future research in designing low-power, high-performance arithmetic units for energy-efficient digital systems.

In an era dominated by mobile computing, IoT devices, and increasingly complex algorithms, the demand for energy-efficient and area-optimized hardware is relentlessly growing. Multipliers, fundamental building blocks in digital signal processing (DSP), image processing, machine learning, and various arithmetic operations, are often power-hungry and area-intensive, particularly for high-precision computations. This inherent characteristic makes them a prime target for optimization. Approximate computing has emerged as a powerful paradigm to address this challenge. It leverages the inherent error resilience of many applications to trade off computational accuracy for significant gains in power consumption, area footprint, and often, delay. This is especially relevant in applications where perfect accuracy

is not strictly necessary, and a slight degradation in output quality is acceptable, such as in multimedia processing (image and video), recognition tasks, and certain types of data analysis. Within the realm of multipliers, the Wallace Tree Multiplier (WTM) stands out as a high-speed architecture, renowned for its efficient reduction of partial products. However, even with its optimized structure, the standard WTM can still be complex and power-consuming, especially for larger bit widths. This creates a compelling motivation for exploring approximate versions of the Wallace Tree Multiplierthe Approximate Wallace Tree Multiplier (AWTM).

Partial Product Generation: For two n-bit operands (A and B), the first step in multiplication is generating n x n partial products. Each partial product is the result of multiplying a bit of operand A with operand B. Partial Product Reduction: The key innovation of the Wallace Tree lies in its efficient method of reducing these partial products. It employs a multi-stage reduction process using carry-save adders (CSAs), often implemented as full adders (FAs).Reduction Stages: In each stage, groups of three partial product rows are reduced to two rows using CSAs. This process is repeated iteratively. The Wallace Tree aims to reduce the number of rows to two as quickly as possible. Irregular Structure: Unlike array multipliers, the Wallace Tree has a more irregular and tree-like structure. This irregularity stems from the optimal grouping of partial products at each reduction stage, leading to a faster reduction process and hence, higher speed compared to array multipliers, especially for larger bit widths. Final Adder: Once the Wallace Tree reduction stages have yielded two rows of partial products, a fast carry-propagate adder (CPA), such as a carry-lookahead adder or a Kogge-Stone adder, is used to sum these two rows and produce the final product.

High Speed: The Wallace Tree architecture is inherently faster than array multipliers, particularly for larger bit widths, due to its optimized partial product reduction. Logarithmic Reduction Time: The number of reduction stages in a Wallace Tree grows logarithmically with the number of bits, contributing to its speed advantage.

Complexity and Area: Despite its speed, the Wallace Tree is more complex than simpler multipliers like array multipliers, requiring a significant area footprint, especially as bit-width increases. The irregular routing also contributes to area overhead. Power Consumption: The complexity and the numerous full adders involved in the reduction stages contribute to considerable power dissipation.

AWTMs seek to alleviate the complexity, area, and power consumption of standard WTMs by introducing approximations at various levels of the multiplier architecture. Here are the primary techniques employed: Selective Partial Product Elimination: One of the most direct ways to approximate a WTM is by strategically eliminating some of the less significant partial products. This reduces the number of inputs to the Wallace Tree reduction stages, simplifying the subsequent addition process.Least Significant Bit (LSB) Partial Product Truncation: Partial products corresponding to the LSBs are often targeted for removal as they contribute less to the overall magnitude of the result and are more likely to be tolerable for approximation. The number of LSB partial products truncated determines the level of approximation and the resulting area and power savings. Column-Based Truncation: Instead of individually selecting partial products, entire columns of partial products in the lower significance positions can be removed. This is easier to implement and control.

Reduces the number of adders and interconnections in the Wallace Tree, directly lowering area and power. However, it also introduces error, primarily in the lower bits of the output, impacting accuracy.

Replacing Accurate Adders with Approximate Adders: The reduction stages of a WTM heavily rely on full adders (FAs). A key approximation technique is to substitute some or all of these FAs with approximate adders.Simplified versions of full adders that are less complex

and consume less power but may produce slightly inaccurate sums under certain input combinations. Examples include: Simplify the carry chain in the adder, potentially ignoring carry propagation beyond a certain point. In LOA, the carry output is approximated using simpler logic (like OR gates) instead of the more complex XOR/AND logic in a full adder. Simplify the carry-skip logic to reduce hardware. These adders are intentionally designed to be inaccurate under specific input conditions to achieve significant power and area reduction. The inaccuracies are often statistically characterized and designed to be within acceptable error bounds.

Significant power and area reduction are achieved by using simpler adders. The error introduced depends on the type and number of approximate adders used. Careful selection and placement of approximate adders within the Wallace Tree are crucial to manage the error.

Using Approximate Carry-Propagate Adders (CPAs): The final adder, responsible for summing the last two rows from the Wallace Tree, is also a target for approximation. Similar to the adders within the reduction stages, CPAs can be replaced with approximate CPAs. While RCA is slower than other CPAs, an approximate design might use a simplified RCA with fewer stages or approximate full adders within the RCA stages. However, focusing on speed is often a motivation for WTM, so drastically slowing down the final adder might negate the benefits. CLAs can also be approximated by simplifying the carry generation and propagation logic, reducing complexity at the cost of accuracy.

Reduces the complexity and power of the final summation. The impact on accuracy depends on the chosen approximate CPA design.

Combining Partial Product Reduction and Approximate Adders: For maximum power and area gains, AWTM designs often employ a combination of techniques. For instance, they might truncate some LSB partial products and use approximate adders in the reduction stages and/or the final adder. More advanced AWTMs might incorporate adaptive approximation, where the level of approximation is dynamically adjusted based on input data characteristics or application requirements. This can provide a balance between accuracy and energy efficiency.

Significant Power Reduction: By simplifying the structure and using less complex components, AWTMs can achieve substantial power savings compared to standard WTMs. The power reduction is typically proportional to the level of approximation introduced. Reduced Area Footprint: The simplification of the reduction stages and potential truncation of partial products lead to a smaller chip area, which is crucial in area-constrained applications like mobile devices and embedded systems. Potentially Improved Speed (in some cases): While the primary focus is often power and area, in certain AWTM designs, simplification of critical paths can also lead to slight improvements in speed, although this is not always the primary goal. Tunable Accuracy-Power/Area Trade-off: AWTMs offer a design space where the level of approximation can be adjusted to meet specific application requirements. Designers can choose the approximation techniques and parameters to achieve the desired balance between accuracy and power/area.

Error Rate Metrics:Mean Error Distance (MED): Measures the average absolute difference between the outputs of the approximate multiplier and the accurate multiplier. Normalized Error Distance (NED): Normalizes the MED by the maximum possible output value to provide a relative error measure. Root Mean Square Error (RMSE): Calculates the square root of the mean of the squared errors, providing another measure of the magnitude of errors. Image Quality Metrics: When AWTMs are used in image processing, it is essential to evaluate their impact on image quality using metrics that correlate with human perception.Peak Signal-to-Noise Ratio (PSNR): A widely used metric that measures the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Higher PSNR generally indicates better image quality. However, PSNR does not

always perfectly correlate with perceived quality. Structural Similarity Index (SSIM): SSIM is considered a more perceptually relevant metric than PSNR. It measures the similarity between two images by considering luminance, contrast, and structural information. SSIM values closer to 1 indicate higher similarity and better perceived quality. Image Fidelity Metrics (e.g., Feature Similarity Index (FSIM)): More advanced metrics like FSIM attempt to capture image features more directly and correlate even better with human visual perception.

Lena Image as a Benchmark: The "Lena" image (often, "Lena Soderberg") is a classic test image in image processing and is frequently used to visually assess the impact of approximations in image processing pipelines. Visual Artifact Analysis: When using AWTMs in image processing tasks (e.g., image filtering, transforms, etc.), the output images are visually inspected for artifacts introduced by the approximation. Common artifacts include:Blockiness: Appearances of blocks, especially in smooth gradient regions, due to errors in computations. Blurring: Loss of sharpness and fine details due to approximation. Color Distortions: Incorrect color rendering if approximation affects color channels differently. Noise Amplification: In some cases, approximation can amplify existing noise in the image. Subjective Assessment: In addition to quantitative metrics, subjective assessment by human observers is often performed to judge the visual acceptability of the approximated images. This involves showing original and approximated images to viewers and asking them to rate the quality or detect differences.

Application-Specific Requirements: The acceptable level of accuracy depends heavily on the application. For some applications (e.g., certain types of multimedia processing), a noticeable degradation might be acceptable for significant power savings. For other applications requiring higher fidelity, the level of approximation must be carefully controlled. Accuracy vs. Power/Area Curve: Designers typically aim to characterize the trade-off curve between accuracy (measured by metrics like PSNR, SSIM) and power/area for different AWTM designs. This helps in selecting the AWTM that best meets the application's constraints and requirements.

Focus on Delay Optimization: ADMs are often designed with a primary focus on reducing delay (latency) while also aiming for power and area efficiency. They achieve this often by simplifying the critical path of the multiplier. Example Techniques in ADMs:Carry-Select Adder Based Approximation: ADMs might use simplified or approximate carry-select adders in their architectures, aiming to speed up carry propagation and overall computation. Pipeline Optimization with Approximation: ADMs might aggressively pipeline the multiplier stages and use approximation within each stage to maintain high throughput while reducing latency and power. AWTM Strength: AWTMs are often more explicitly targeted at power and area reduction, leveraging the Wallace Tree structure's inherent efficiency and applying approximation techniques within its reduction process. ADM Strength: ADMs can be more effective in scenarios where minimizing latency is a critical performance metric, in addition to power and area. Overlap: There can be overlap in techniques. For example, both AWTMs and ADMs might use approximate adders, but the overall architectural focus and optimization goals might differ.

Truncated Multipliers: A more basic form of approximation where some of the least significant partial products are simply discarded. Simpler to implement than AWTMs but can have less controlled error characteristics. Segmented Multipliers: Divide the multiplication into segments and approximate computations within each segment to reduce complexity. Logarithmic Multipliers: Approximate multiplication using logarithmic and anti-logarithmic operations. Can be highly power-efficient for certain precision requirements but often have limited accuracy and may be less directly comparable to bit-parallel multipliers like WTM. Neural Network-Based Approximate Multipliers: Emerging research explores using neural networks

to design approximate multiplier blocks, learning approximation patterns from data

Accuracy Needs: How much error is tolerable? Is visual quality or numerical precision more critical? Performance Goals: Is power consumption the primary concern, or is delay (latency) equally or more important? Area constraints? Implementation Complexity: Some approximation techniques are simpler to implement and control than others. Technology and Design Constraints: The optimal choice might also depend on the target technology (e.g., 90nm, newer technologies) and existing design methodologies.

Relevance to 90nm: In 90nm and similar technologies, power dissipation was a major challenge. Leakage power became more prominent as transistors scaled down, making power-efficient designs crucial. AWTMs provided a valuable approach to manage power in complex arithmetic units like multipliers in these technologies. Continued Relevance in Modern Technologies: While process technology has advanced, the fundamental principles of approximate computing and the benefits of AWTMs remain highly relevant, even more so in modern nanoscale technologies.Increased Transistor Density: Modern technologies allow for even higher transistor densities, enabling more complex systems on a chip, but power density remains a critical constraint. Battery-Powered Devices: The proliferation of mobile devices, wearables, and IoT devices makes power efficiency even more paramount. Emerging Applications: Applications like machine learning, edge computing, and AI in resource-constrained environments further drive the need for energy-efficient approximate computing solutions.

Dynamic and Adaptive AWTMs: Research continues on developing AWTMs that can dynamically adjust their level of approximation based on input data, operating conditions, or application demands. This can lead to even better energy efficiency while maintaining acceptable accuracy when needed. Automated Design and Optimization: Tools and methodologies for automated design space exploration and optimization of AWTMs are crucial to make these techniques more readily accessible to designers. Error Analysis and Characterization: More sophisticated methods for analyzing and characterizing the error behavior of AWTMs are needed to provide guarantees on accuracy and enable more reliable deployment in various applications. Application-Specific AWTM Design: Tailoring AWTM designs to specific application domains (e.g., image processing algorithms, neural network architectures) can lead to further performance improvements. Integration with System-Level Approximation Techniques: Combining AWTMs with other approximate computing techniques at higher levels of abstraction (e.g., algorithm-level approximation, memory approximation) can create synergistic power efficiency gains.

Approximate Wallace Tree Multipliers (AWTMs) represent a significant advancement in power-efficient arithmetic design. By strategically introducing approximations within the Wallace Tree architecture, AWTMs offer a compelling trade-off between accuracy and power/area consumption. They are particularly valuable in applications like image processing where a slight degradation in accuracy is often visually imperceptible or acceptable for the sake of significant energy savings. Continued research into new approximation techniques, dynamic adaptation, and automated design methodologies promises to further enhance the capabilities and applicability of AWTMs in future computing systems, especially in the face of ever-increasing demands for energy efficiency and computational power. This detailed literature review has explored the key aspects of AWTMs, from their foundational principles in the standard Wallace Tree to the diverse approximation techniques, accuracy evaluation methods, and comparisons with other approximate multiplier approaches. The field of approximate computing, and AWTMs in particular, remains a vibrant area of research and development with significant potential to shape the future of energy-efficient digital systems.

4 Theory

4.1 Wallace Tree Multiplier

A Wallace tree multiplier is a fast hardware implementation of binary multiplication, typically used in digital circuits. It is an efficient structure for carrying out multiplication by reducing the number of partial product rows in a parallelized manner. The primary advantage of the Wallace tree over other multiplication architectures, like the traditional array multiplier, is its ability to reduce the number of adders required for partial product summation. This results in faster multiplication with less delay.

In a Wallace tree, the partial products are summed using a tree-like structure of full adders and half adders, which reduces the time complexity of the operation. The structure works by performing pairwise additions of the partial products at each stage, progressively reducing the number of rows of partial products until only two rows remain. A final carry-save adder (CSA) or ripple-carry adder (RCA) is used to produce the final result.

4.2 Approximate Computation

Approximate computing has emerged as a promising technique for reducing the power, area, and delay of digital systems, especially in applications where exact precision is not a strict requirement. The idea behind approximate computing is to trade off accuracy for improvements in performance metrics. In the context of multipliers, approximate multipliers use techniques like truncation, rounding, or using approximate adders to reduce the complexity of the operations.

For error-resilient systems, small inaccuracies in computations may not significantly affect the overall system performance. As a result, using approximate components in the arithmetic units can lead to substantial gains in power consumption and area, which is essential in many modern applications like signal processing, machine learning, and image processing, where slight errors are tolerable.

4.3 Approximate Adder

An approximate adder is a key component in an approximate multiplier. By modifying the conventional adder, which is used to sum the partial products in a Wallace tree, one can reduce the number of bits required to represent the sum or introduce simplified logic for summation. This trade-off reduces the power and area consumption of the circuit. Several methods exist for designing approximate adders, such as:

- **Truncation:** Ignoring certain lower-order bits in the sum, leading to a reduced number of bits being considered.
- **Rounding:** Rounding the sum to the nearest value based on a set of constraints.
- **Logic Simplification:** Replacing full adders with simpler logic gates, which introduces approximation but reduces complexity.

The error of an approximate full adder (approxFA) depends on the entries of the truth table that deviate from the exact full adder. To quantify this error, several error metrics are commonly used:

- **Sum Approximation (Sapp):** Defined for each of the eight entries in the truth table as:

$$S_{app} = 2 \cdot C_{out} + S \tag{1}$$

- **Error Calculation:** The difference between the exact and approximate sum is given by:

$$Err_i = S - S_{app} \tag{2}$$

A positive or negative value of Err_i represents the deviation caused by the approximate full adder.

- **Sum of Absolute Errors (SAE):** Measures the total deviation across all truth table entries:

$$SAE = \sum_{i=0}^{7} |Err_i| \tag{3}$$

The SAE value ranges from 0 (for an exact full adder) to 18 (for an approximate full adder with completely negated entries).

- **Mean Absolute Relative Error (MRED):** Indicates the average deviation relative to the exact output:

$$SAE = \sum_{i=0}^{7} |Err_i| \tag{4}$$

The SAE value ranges from 0 (for an exact full adder) to 18 (for an approximate full adder with completely negated entries).

$$MRED = \frac{1}{7} \sum_{i=1}^{7} |RErr_i| \tag{5}$$

where

$$RErr_i = 1 - \frac{S_{app}}{S}, \quad \forall i = \{1, \ldots, 7\} \tag{6}$$

This metric excludes cases where the input is $\{0,0,0\}$, as errors in this case would lead to an infinite relative error.

- **Error Probability (P_E):** The likelihood of an incorrect output, independent of error magnitude:

$$SAE = \sum_{i=0}^{7} |Err_i| \tag{7}$$

The SAE value ranges from 0 (for an exact full adder) to 18 (for an approximate full adder with completely negated entries).

$$P_E = \frac{1}{8} \sum_{i=0}^{7} (1 if Err_i \neq 0 else 0) \tag{8}$$

The higher the error probability, the greater the chance of an incorrect sum.

- **Mean Error (ME):** Represents the average signed error over all inputs:

$$ME = \frac{1}{8} \sum_{i=0}^{7} Err_i \tag{9}$$

It is useful in cases where errors can be compensated in later processing stages.

- **Maximum Absolute Error (MaxAE):** The worst-case deviation among all possible inputs:

$$MaxAE = \max(|Err_i|), \quad i = \{0, \ldots, 7\} \tag{10}$$

For example, an approximate full adder with three erroneous truth table entries has an error probability of:

$$P_E = \frac{3}{8} = 0.375, \quad ME = -0.25, \quad MaxAE = 2. \tag{11}$$

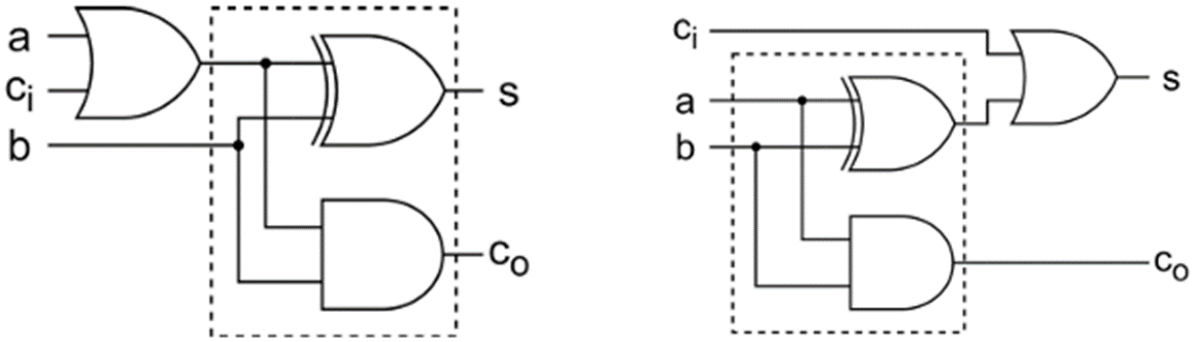


Figure 2: (a) c836 Approximate adder diagram [2], (b) 88fc Approximate adder diagram [2]

Inputs			**FA**		**AFA ONE**		**AFA TWO**	
A	B	C_{in}	Sum	C_{out}	Sum	C_{out}	Sum	C_{out}
0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0
0	1	1	0	1	0	1	0	1
1	0	0	1	0	1	0	1	0
1	0	1	0	1	**1x**	**0x**	**1x**	**0x**
1	1	0	0	1	0	1	**1x**	**0x**
1	1	1	1	1	**0x**	1	1	1

Table 1: Truth table for actual FA and approximate adders C836 and 88F6

5 Methodology

5.1 Implementation of Wallac Tree

The Wallace Tree Multiplier (WTM) is implemented to perform high-speed multiplication by reducing the number of partial product rows using a tree-like structure of full adders and half adders. The methodology follows these steps:

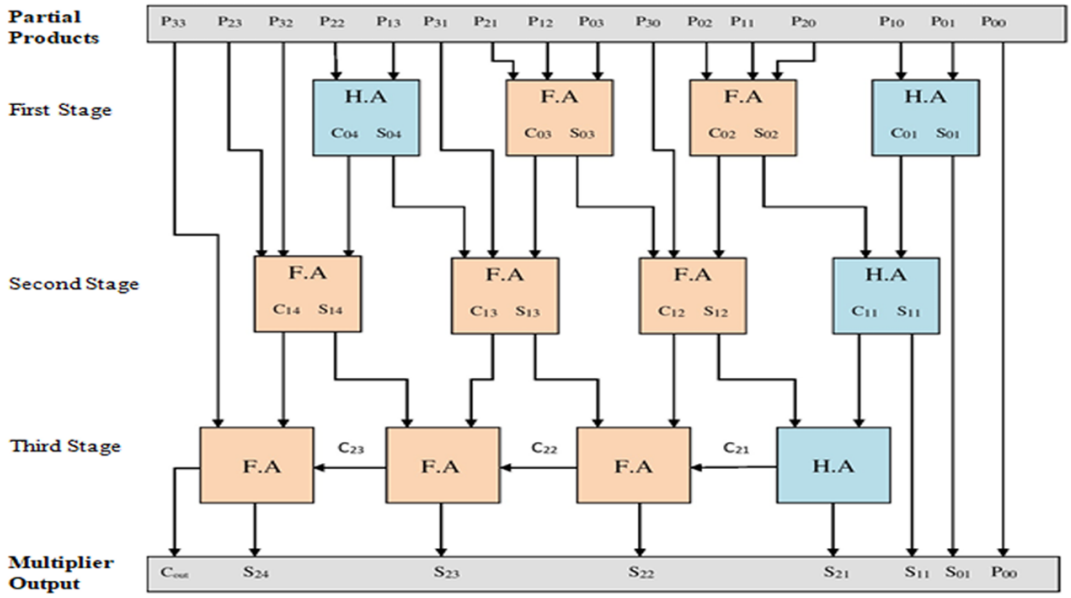


Figure 3: 4×4 Wallace Tree block diagram [3]

- **Partial Product Generation:** The multiplication process begins with generating partial products using AND gates. For an $N \times N$ multiplication, N^2 partial product bits are generated.
- **Reduction using Wallace Tree:** The partial products are summed in a parallel manner using full adders (FA) and half adders (HA). The Wallace tree structure performs bitwise reduction by grouping three rows at a time and reducing them until only two rows remain.

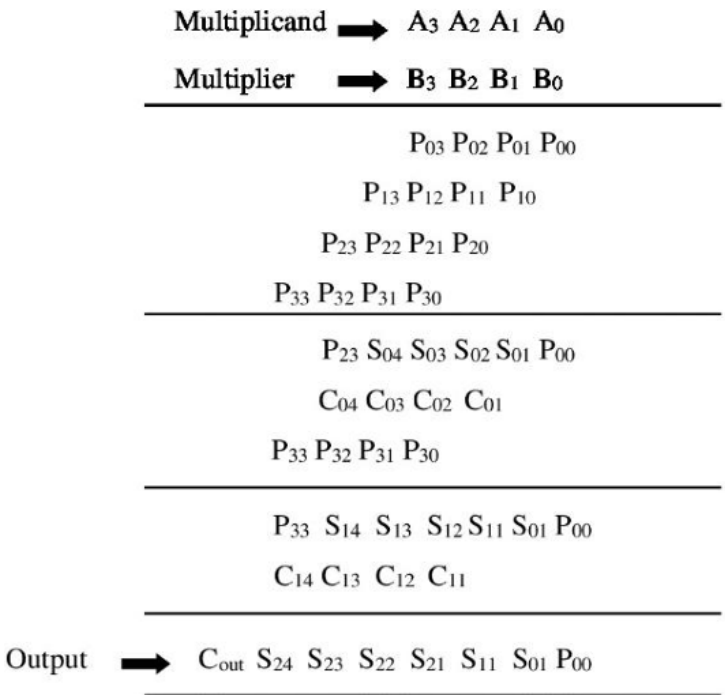


Figure 4: Operation of 4×4 Wallace Tree Multiplier [3].

- **Final Summation:** The last two rows are added using a conventional ripple carry adder

(RCA) or carry-lookahead adder (CLA) to obtain the final product. This method significantly reduces the delay compared to traditional array multipliers.

To improve the power and area efficiency of the Wallace Tree Multiplier, full adders in the reduction stage are replaced with approximate full adders. The modifications are as follows:

- **Selection of Approximate Adder:** A low-power approximate full adder is chosen based on its error resilience and efficiency in reducing power and delay. The selected approximate adders trade accuracy for power and area savings.
- **Integration in Wallace Tree:** The Wallace tree reduction stage is modified by substituting exact full adders with approximate full adders. This replacement reduces the number of transistors and switching activity, leading to power and area savings.
- **Analysis of Performance Impact:** The impact of approximation is evaluated by measuring power, delay, and error metrics. The output error is assessed to ensure the approximation remains within acceptable limits, especially for applications like image processing and signal processing.

By integrating approximate adders, the modified Wallace Tree Multiplier achieves a balance between computational accuracy and hardware efficiency, making it suitable for error-resilient applications.

5.2 Implementation of DCT with Approximate Adder

To evaluate the impact of approximate computing, the Discrete Cosine Transform (DCT) is implemented using both an exact and an approximate adder. The approximate adder modifies the summation process to reduce power and area consumption while introducing minor computational errors.

The following Python code implements both the exact and approximate DCT:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.metrics import peak_signal_noise_ratio as psnr
from skimage.metrics import structural_similarity as ssim

# Exact Adder Function
def full_adder(a, b, c_in):
    sum = (a^b)^c_in
    c_out = (a & b) | (a^b) & c_in
    return sum, c_out

# EC36 Approximate Full Adder
def approx_full_adder(a, b, c_in):
    sum = (a & ~b) | (~b & c_in) | (~a & b & ~c_in)
    c_out = (a & b) | (b & c_in)
    return sum, c_out

# Apply Approximate Adder to DCT Coefficients
def apply_approximate_adder(dct_matrix):
    rows, cols = dct_matrix.shape
    approx_dct = np.zeros_like(dct_matrix, dtype=np.float32)

```

```
    for i in range(rows):
        for j in range(cols):
            sum_, _ = approx_full_adder(int(dct_matrix[i, j]), 0, 0)
            approx_dct[i, j] = sum_

    return approx_dct

# Load Image and Compute DCT
image = cv2.imread("input_image_lena.jpg", cv2.IMREAD_GRAYSCALE)
image_float = np.float32(image)

exact_dct = cv2.dct(image_float)
approx_dct = apply_approximate_adder(exact_dct)

# Compute Inverse DCT and PSNR/SSIM
lena_exact_idct = cv2.idct(exact_dct)
lena_approx_idct = cv2.idct(approx_dct)

# Convert to uint8 for Display
lena_exact_idct = np.uint8(np.clip(lena_exact_idct, 0, 255))
lena_approx_idct = np.uint8(np.clip(lena_approx_idct, 0, 255))

psnr_approx = psnr(image, lena_approx_idct, data_range=255)
ssim_approx = ssim(image, lena_approx_idct, data_range=255)

plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(lena_approx_idct, cmap='gray')
plt.title(f"Approx DCT (PSNR: {psnr_approx:.2f}, SSIM: {ssim_approx:.3f
    })")
plt.axis("off")

plt.show()
```

Listing 1: Python Code for Approximate DCT

This implementation allows us to evaluate the performance of the approximate adder in DCT transformation.

6 Proposed Design

In this section, we present the detailed layout and structural design of various logic gates, adders, and multipliers used in our proposed approximate computing approach. The focus of our design is on optimizing power consumption, reducing area overhead, and maintaining computational accuracy within acceptable limits.

6.1 Logic Gate Layouts

To construct efficient arithmetic units, we have designed fundamental logic gates, which serve as the building blocks for more complex arithmetic circuits such as adders and multipliers. The

layouts of the XOR gate, AND gate, and OR gate are shown below. These gates are essential components for constructing the arithmetic logic units used in the proposed design.

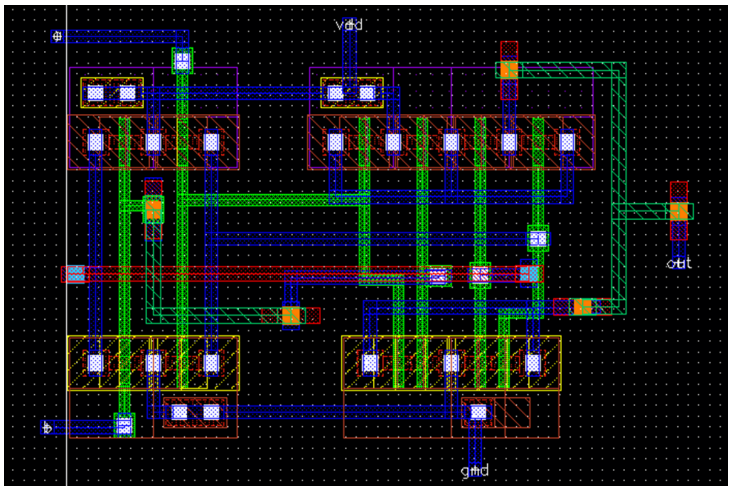
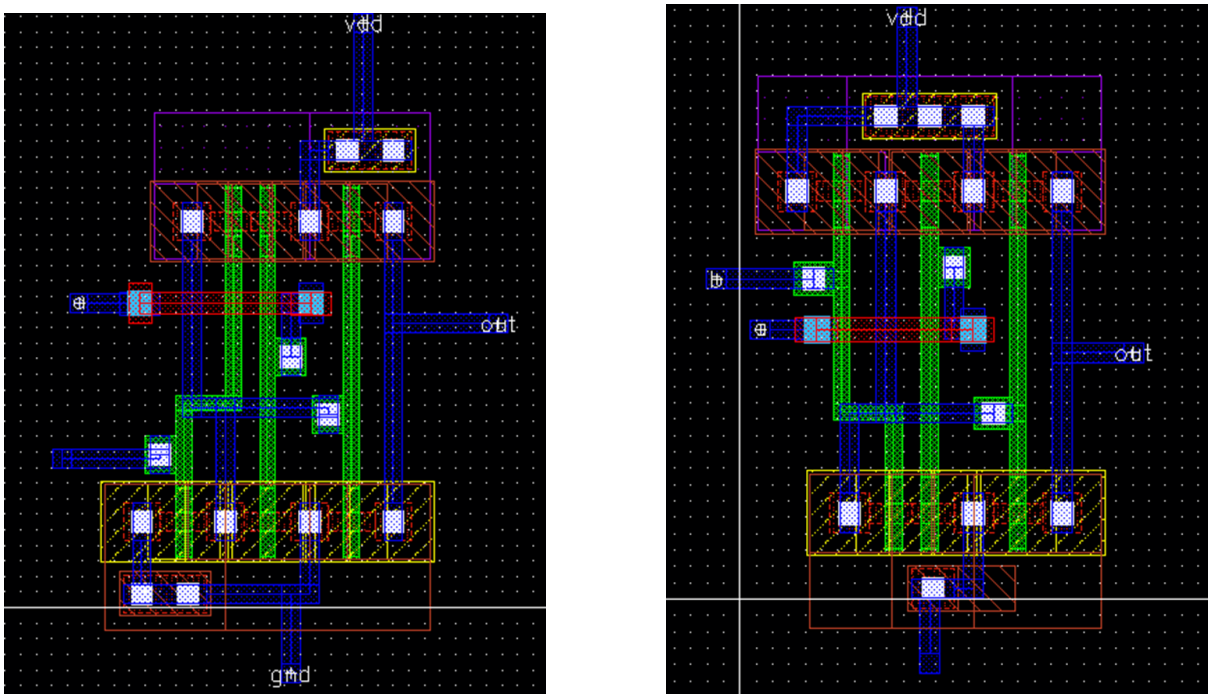


Figure 5: Layout of X-OR gate



(a) (b)

Figure 6: (a): Layout of And gate, (b): Layout of OR gate.

6.2 Approximate Full Adders

The approximate computing paradigm introduces modifications to traditional full adders to achieve a balance between computational accuracy and hardware efficiency. In our study, we employ two approximate full adders:

- C836 Approximate Full Adder

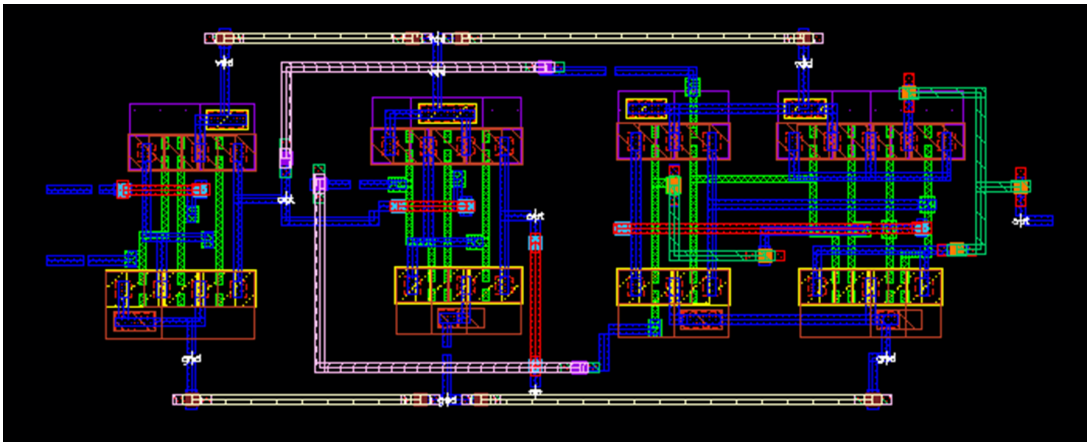


Figure 7: Layout of Approximate Full Adder (C836)

- 88F6 Approximate Full Adder

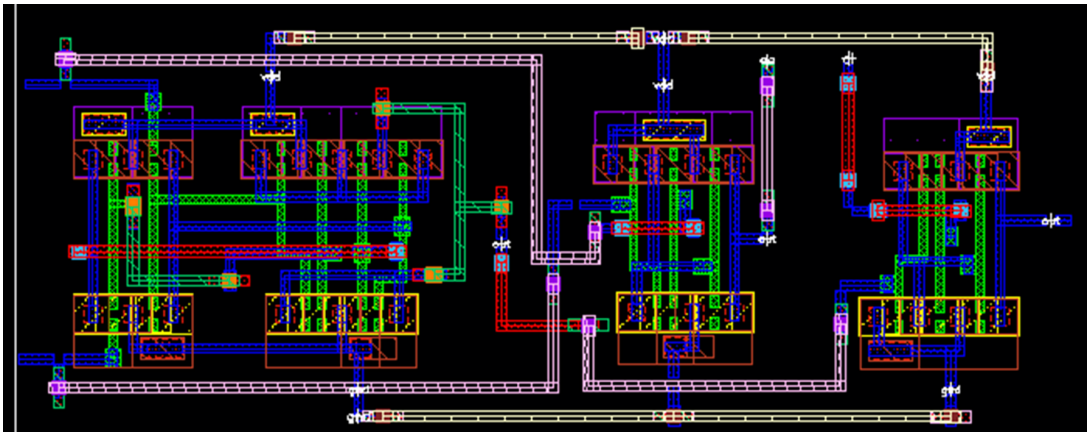


Figure 8: Layout of Approximate Full Adder (88f6)

Both of these adders reduce power and area consumption while introducing a tolerable error in computation.

6.3 Half and Full Adder Layouts

The primary components of arithmetic units are half and full adders, which perform binary addition operations. The layouts of these components define the transistor-level design that ensures efficient signal propagation and minimizes power dissipation.

6.4 Wallace Tree Multiplier with Approximate Adders

A Wallace Tree Multiplier (WTM) is a high-speed multiplication circuit that optimizes partial product summation. To further enhance efficiency, we introduce approximate adders into the Wallace tree structure. Two variations of the Approximate Wallace Tree Multiplier (AWTM) are presented:

- Wallace Tree Multiplier using the C836 Approximate Full Adder
- Wallace Tree Multiplier using the 88F6 Approximate Full Adder

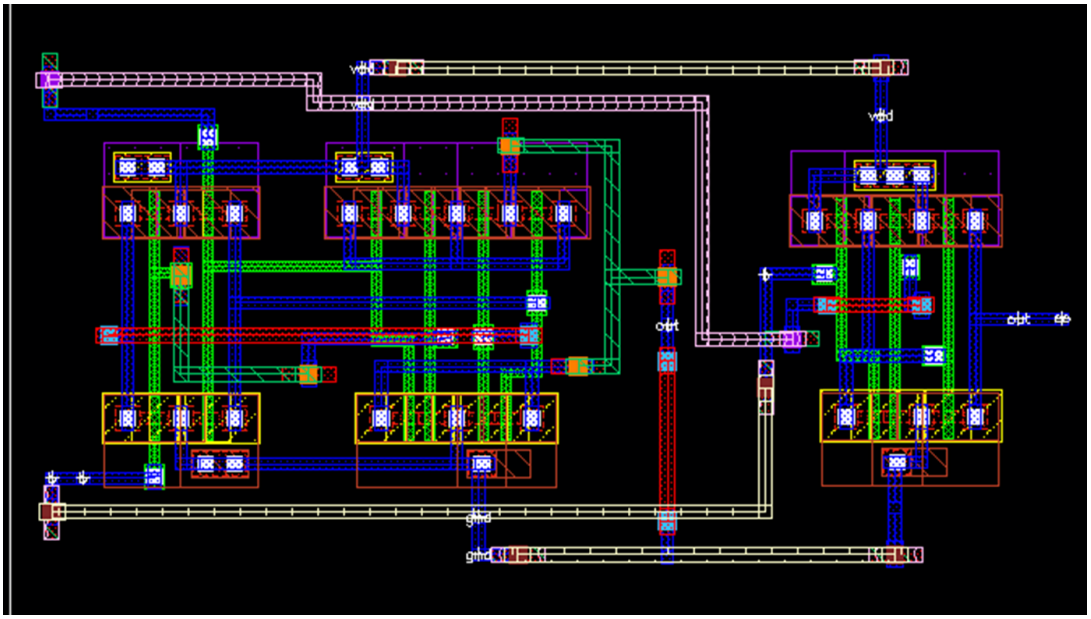


Figure 9: Layout of Half Adder

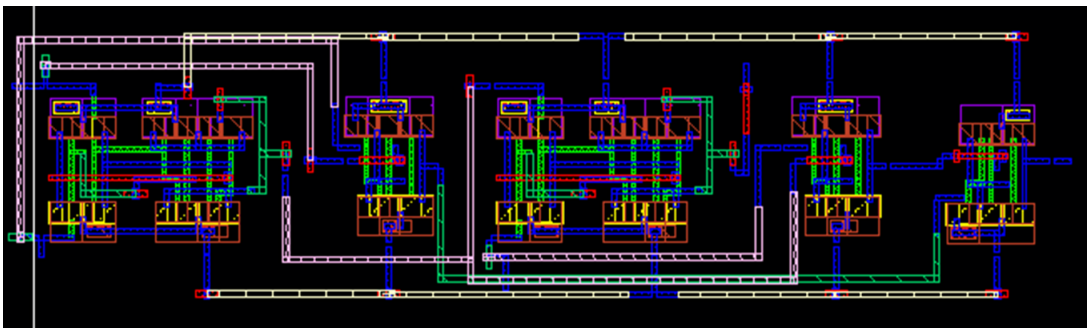


Figure 10: Layout of Full Adder

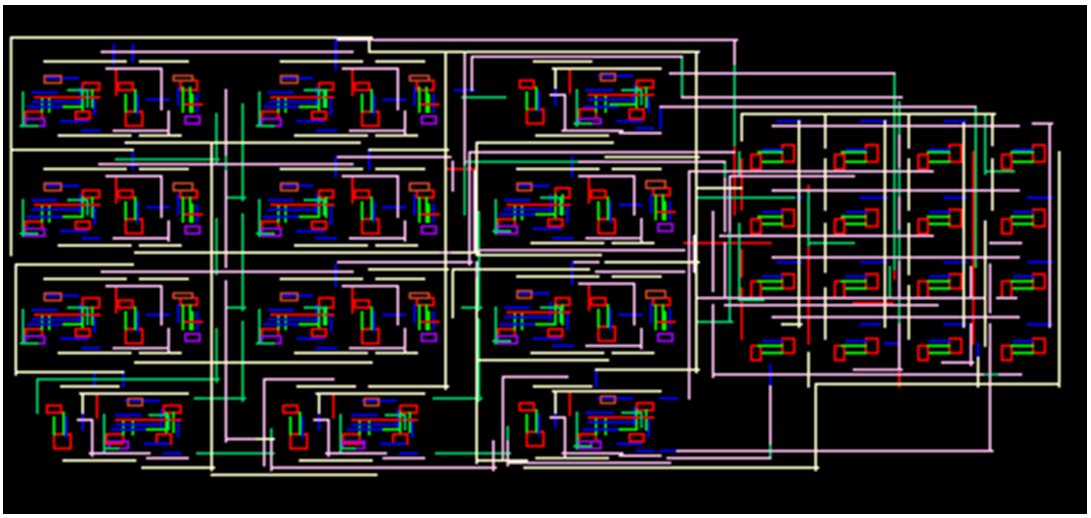


Figure 11: Layout of Approximate Wallace Tree Multiplier(Using c836 approx adder)

By replacing conventional full adders with approximate versions, we achieve significant reductions in power consumption and area while maintaining a reasonable level of accuracy.

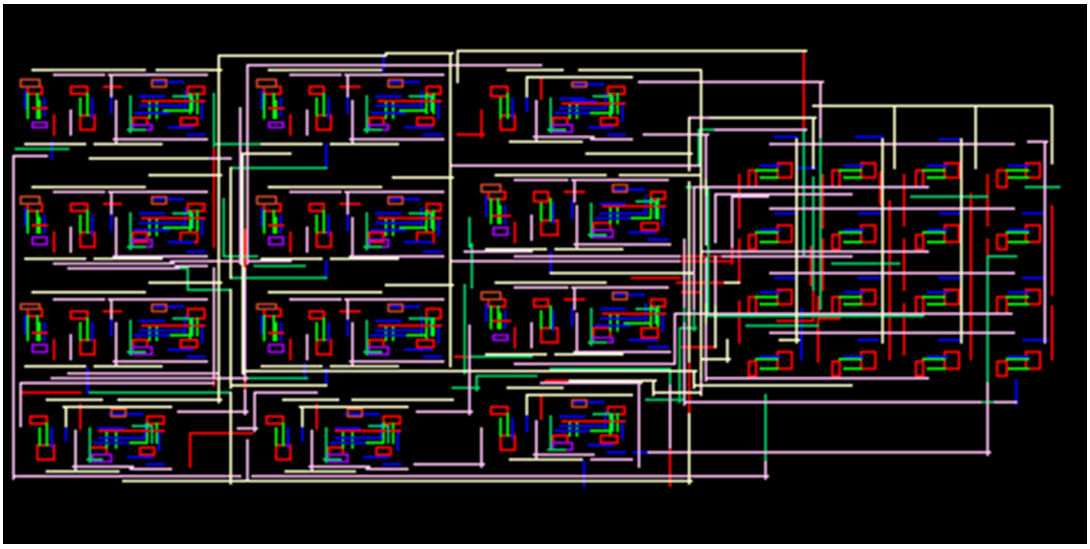


Figure 12: Layout of Approximate Wallace Tree Multiplier(Using 88f6 approx adder)

7 Simulation Result

7.1 Wallace Tree Multiplier Performance

Parameter	WTM	AWTM (C836)	C836 Post-layout	AWTM (C876)	C876 Post-layout
Power (μW)	17.4	12.89	N/A	12.92	29.64
Delay (ps)	225.65	110.4	N/A	199.225	813.7
PDP (fJ)	3.92	1.42	N/A	2.57	24.11
Area (μm^2)	106×33	72.5×32.3	N/A	72.2×31.7	–

Table 2: Wallace Tree Multiplier Comparison

The Wallace Tree Multiplier (WTM) and its approximate variants were evaluated in terms of power consumption, delay, and power-delay product (PDP). As shown in Table 2, the conventional WTM consumes 17.4 µW of power, whereas the Approximate Wallace Tree Multiplier (AWTM) with C836 reduces power consumption to 12.89 µW, demonstrating the efficiency of approximation. However, the post-layout AWTM (C876) shows an increase in power consumption to 29.64 µW, likely due to additional interconnect overhead. In terms of delay, the conventional WTM exhibits the highest delay at 225.65 ps, whereas AWTM (C836) reduces it significantly to 110.4 ps. However, the post-layout AWTM (C876) experiences a substantial delay increase to 813.7 ps, likely due to layout constraints. The PDP values further confirm these observations, with WTM having a PDP of 3.92 fJ, AWTM (C836) achieving a more energy-efficient 1.42 fJ, and post-layout AWTM (C876) increasing to 24.11 fJ due to its delay overhead.

In terms of area utilization, the conventional WTM occupies 106×33 µm², whereas AWTM (C836) reduces this to 72.5×32.3 µm², showing that approximation also leads to a reduction in hardware complexity. The post-layout AWTM (C876) has a slightly smaller footprint of 72.2 $\times$ 31.7 µm², indicating minimal variation in area despite the changes in power and delay performance. These results highlight that while approximate multipliers can significantly improve power and area efficiency, their performance in post-layout implementation must be carefully optimized to avoid excessive delay overhead.

7.2 Approximate Adder Performance

Parameter	Exact Full Adder	C836	88F6
Power (nW)	1010	524.7	554.2
Delay (ps)	51.2	58.37	25.11
Area (um)	158.4	75.52	73.17
PDP (atto)	51.712	30.53	13.92
Efficiency(Power)	-	51.9%	54.8%

Table 3: Comparison of Exact Full Adder, C836, and 88F6

The performance of approximate adders is evaluated based on power, delay, and PDP, as shown in Table 3. The exact full adder consumes 1010 nW, whereas the approximate adders C836 and 88F6 significantly reduce power consumption to 524.7 nW and 554.2 nW, respectively. This confirms that approximate computing effectively reduces power dissipation. In terms of delay, the exact full adder has 51.2 ps, while C836 has a slightly higher delay of 58.37 ps.

The 88F6 adder achieves the lowest delay at 25.11 ps, making it suitable for high-speed applications. The exact full adder have power delay product 51.712 attoJ, while C836 and 88F6 achieve significantly lower values of 30.53 attoJ and 13.92 attoJ, respectively, making them efficient power consumption.

The remarkable compact area consumption of two approximate Full adder is ideal for resource constrained application, where C836 and 88F6 significantly reduce area usage to 75.52 μm^2 and 73.17 μm^2, respectively. The exact full adder occupies 158.4 μm^2. Additionally, the power efficiency improvement of C836 and 88F6 is 51.9% and 54.8%, respectively, demonstrating that approximation techniques can achieve substantial gains in power efficiency while maintaining acceptable levels of accuracy.

7.3 Simulation Results of DCT with Approximate Adder

To evaluate the impact of approximate computing on image processing applications, the Discrete Cosine Transform (DCT) was performed using both an exact adder and an approximate adder. The resulting images and their corresponding quality metrics—Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM)—are shown in the figure.

The original image is used as a reference to compare the reconstructed images obtained from both exact and approximate DCT computations. The image processed using the exact adder achieves a PSNR of 51.55 dB and an SSIM of 1.000, indicating minimal distortion and near-perfect structural preservation. On the other hand, the approximate adder yields a PSNR of 49.81 dB and an SSIM of 1.000, which signifies a slight degradation in image quality.

Despite the minor reduction in PSNR, the SSIM values remain at 1.000, suggesting that the perceptual quality of the images remains virtually indistinguishable. This confirms that the use of an approximate adder in DCT computations introduces minimal visual artifacts while reducing power and area overhead.

These results highlight the effectiveness of approximate computing in image processing applications, where a slight reduction in numerical accuracy can lead to substantial improvements in power and delay performance without significantly affecting visual quality.

Figure 13: DCT of an image using Exact adder and Approx Adder

7.4 Post Layout Simulation Results

To evaluate the performance difference between schematic and post-layout designs, we conducted post-layout simulations for two Approximate Wallace Tree Multipliers (AWTMs). The simulations were performed using configuration files that enabled us to analyze the delay and power characteristics comprehensively.

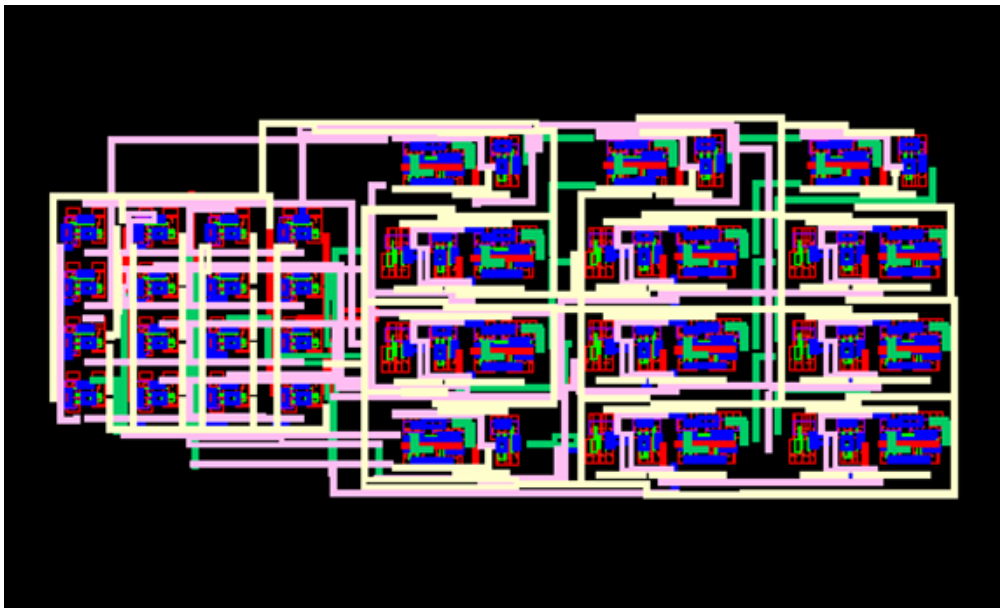


Figure 14: Parasitic extracted layout view of AWTM (One).

For delay analysis, we focused on the Most Significant Bit (MSB) and the output O3**, as these critical outputs provide an accurate representation of the overall timing behavior of the multiplier. By comparing the propagation delays in both schematic and post-layout implementations, we assessed the impact of parasitic elements introduced during the physical layout process. These parasitics, including resistance, capacitance, and interconnect delays, often degrade the performance of the circuit compared to the ideal schematic simulation.

Furthermore, for power consumption analysis, we computed the average power dissipation in both designs. The average power was measured by integrating the power waveform over a complete operational cycle and normalizing it with respect to time. This analysis provided insights into how post-layout factors such as increased wire resistance, coupling capacitance, and signal integrity issues influence the power efficiency of the AWTMs.

By systematically comparing the delay and power characteristics of both implementations, we aimed to quantify the trade-offs associated with layout-level optimizations and provide

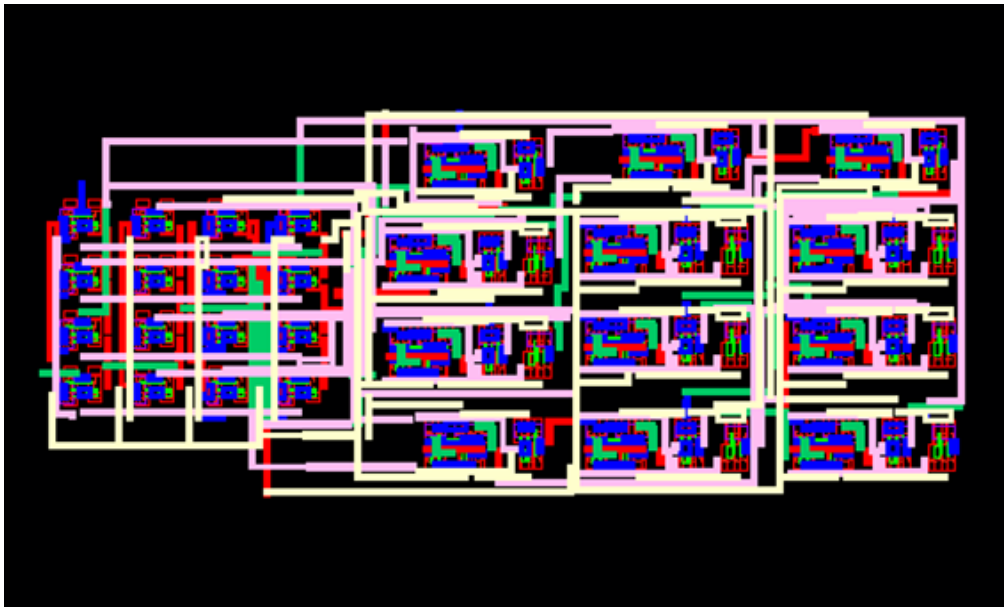


Figure 15: Parasitic extracted layout view of AWTM (two).

guidelines for enhancing the design's performance in real-world fabrication scenarios.

- **Nodes**: 3518
- **BSIM3v3 Transistors**: 360
- **Parasitic Capacitors**: 2680
- **paracisitcResistors**: 3388
- **Voltage Sources**: 9

Figure 16: Circuit Inventory AWTM(One)

Circuit Inventory

- **Nodes**: 3533
- **Transistors**: 360
- **Parasitic Capacitors**: 2774
- **Parasitic Resistors**: 3403
- **Voltage Sources**: 9

Figure 17: Circuit Inventory AWTM(two)

8 Conclusion

In conclusion, this report explores the use of approximate computing to enhance the performance of Wallace Tree Multipliers, particularly within energy-efficient Computing-In-Memory

(CIM) architectures. CIM architectures have gained attention due to the increasing demands of neural networks. Many SRAM-based CIM architectures use digital adder trees (ATs) to perform in-memory multiply-accumulate (MAC) operations. By substituting full-adders in the Wallace tree structure with approximate adder logic, the design achieves significant improvements in power consumption and area efficiency, addressing a key challenge in digital CIM implementations where adder trees incur substantial overhead. The first proposed ap- proximate wallace tree multiplier scheme achieves 33.05% reduction in area and a 25.9% where another one achieves 25.75% reduction in power and 34.57% reduction in area. The effectiveness of approximate adders is validated through the computation of the Discrete Cosine Transform (DCT) on the benchmark image "Lena." The results show a high PSNR (Peak Signal-to-Noise Ratio) of 49.81 dB, indicating that the approximate adder introduces only slight errors, while maintaining a similar SSIM (Structural Similarity Index), thus preserving the essential structural information of the image. This work contributes to the ongoing research in energy-efficient computing by providing a practical approach to designing low-power, high-performance arithmetic units for CIM-based digital systems. The findings highlight the potential of approximate computing to meet the increasing demands for energy efficiency and computational power in various applications, including image processing and other error-resilient domains, by optimizing a critical component in CIM architectures

References

[1] K. Naseri and H. Jahanirad, "A runtime reconfigurable exact-approximate full-adder design," in *2024 6th Iranian International Conference on Microelectronics (IICM)*, pp. 1–5, 2024.

[2] E. Napoli, E. Zacharelos, A. G. Strollo, and G. Di Meo, "Approximate full-adders: A comprehensive analysis," *IEEE Access*, 2024.

[3] M. Esa and K. Achyut, "Design and verification of 4 x 4 wallace tree multiplier," *International Journal of Analytical and Experimental Modal Analysis (IJAEMA)*, vol. 11, no. 10, pp. 657–660, 2019.

[4] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power-and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Fifteenth international symposium on quality electronic design*, pp. 263–269, IEEE, 2014.

[5] C. He, Z. Wang, F. Xiang, Z. Dai, Y. He, J. Yue, and Y. Liu, "Lsac: A low-power adder tree for digital computing-in-memory by sparsity and approximate circuits co-design," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 2, pp. 852–856, 2023.

[6] G. Anusha and P. Deepa, "Design of approximate adders and multipliers for error tolerant image processing," *Microprocessors and Microsystems*, vol. 72, p. 102940, 2020.