SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I3P107

Volume 13 Issue 3, 84-98, March 2026

Original Article

A Low Power Hybrid Bit-Adaptive Approximate Multiplier with LOD Guided Approximation and Fault-Tolerant Adders for Image Processing Applications

Lokhanadam Vara Prasad[1], K. Venkata Ramanaiah[2]

[1,2]*Department of Electronics and Communication Engineering, YSR Engineering College of Yogi Vemana University, Proddatur, Andhra Pradesh, India.*

[1]*Corresponding Author : varaprasadlokhanadam@gmail.com*

Received: 07 December 2025 Revised: 09 January 2026 Accepted: 08 February 2026 Published: 23 March 2026

Abstract *- Energy-constrained digital signals, image-processing platforms, and machine-vision systems rely heavily on multipliers owing to their crucial role in managing power, area, and computation speed. Traditional high-precision multipliers suffer from heavy latency and power consumption caused by complex partial product generation and carry propagation paths, which are unsuitable for low-power real-time operation. To overcome these drawbacks, this paper proposes a Hybrid Bit-Adaptive Approximate Multiplier (HBAM), which adaptively varies its computing precision according to the prominent value inside the inputs by adopting an effective Leading-One Detection (LOD) scheme, a fault-tolerant adaptive adder. The architecture minimizes unnecessary switching activity and redundant computation by bit-adaptive truncation involving partial protection of high-order bits, and selects a set of high-precision significant bits for accuracy. The implementation results demonstrate that the proposed HBAM achieves substantial improvements over state-of-the-art approximate multipliers. For 8-bit designs, HBAM achieved the lowest area (610 µm²), lowest power (0.59 mW), and fastest delay (0.91 ns). The 16-bit solution also reports about 2,080 um (2) area, 1.68 mW power, and 1.59 ns delay over competing structures such as TOSAM, PPC, AMA-x, and LOA by a factor of even orders of magnitude. Application-level evaluation on standard benchmark images also demonstrates that the proposed HBAM yields better smoothing and edge detection quality, with a PSNR up to 40.00 dB and SSIM of 0.995 for smoothing and a PSNR up to 33.50 dB/ SSIM of 0.940 for edge detection, respectively, all at low arithmetic error (MSE of down to 6.50). These results show that the proposed HBAM architecture provides an excellent tradeoff between low power consumption, high performance, and high image quality and is highly suited for next-generation low-power image processing and embedded vision systems.*

Keywords *- Approximate Computing, Hybrid Bit-Adaptive Multiplier (HBAM) and Leading-One Detection (LOD), Image Processing, Edge Detection and Smoothing Filters; Energy-Efficient Multiplier Architecture*.

1. Introduction

Approximate computing has become a key strategy for designing low-power and high-performance arithmetic circuits, particularly in error-tolerant domains, such as image processing, deep learning, and embedded systems. Because multipliers are one of the most power-hungry modules in contemporary Integrated Circuits (ICs), a wealth of literature has focused on designing approximate multipliers that can compromise computing accuracy and hardware efficiency. Most recent studies have designed error-compensated and precision-aware architectures to reduce quality losses and achieve significant reductions in area, power, and delay [1, 2]. Area-efficient and image-focused approximate multiplier designs have further demonstrated that controlled approximation can enhance processing efficiency without noticeably affecting the visual quality [3, 5]. Highly accurate approximate compressors and hybrid approaches have been proposed to enhance the accuracy and structural efficiency of multiplier designs [4, 7]. Analogously, encoded partial products, inexact counters, and architecture-aware optimizations have been applied to improve the performance of multimedia applications such as JPEG compression [6], noise-resilient filtering [8], and recursive low-power image multipliers [9]. Approximate multipliers have been introduced in emerging AI systems for fault tolerance, adaptive computation timing error resilience to support lightweight and energy-efficient inference engines [10, 13]. To further decrease hardware complexity, small and energy-efficient approximate multiplier blocks [12] and optimized adder structures such as significant-input extraction and reversible

adders have also been proposed to implement error-tolerant computing [14, 15]. Approximate compressors designed explicitly for image-processing-type workloads have high accuracy but require less gate count and power consumption [16].

Furthermore, scalable and truncation-based approximation methods [19], compressor-controlled unsigned multipliers [20], and booth-encoded low-power structures [21] extend the design options for bit-adaptive computation. In addition to CMOS techniques, approximate computing has been applied to Neuromorphic Systems and Memristor-Based matrix multipliers to accelerate Convolutional Neural Networks [17, 18]. Fast emulation efforts are also aiding the assessment of strategies developed for approximate computing in DNN settings [22]. These schematic gaps motivate the demand for hybrid adaptive approximation schemes that tune the precision depending on the importance of an input.

Motivated by these advancements, the proposed Hybrid Bit-Adaptive Approximate Multiplier (HBAM), with LOD-guided approximation and fault-tolerant adder structure, is intended to greatly enhance energy efficiency and hardware complexity, as well as robustness against modern image-processing applications. The adaptation in the architecture is performed at the bit level, where the computational precision of an operand is dynamically set based on its significance, leading to low switching activity in less significant parts of the computation.

The integrated Leading-One Detection (LOD) methodology is capable of facilitating real-time approximate control by detecting leading bit positions directly and limiting approximations whenever the top bits are high, such that the output accuracy level is ensured. To this end, the proposed designs incorporate a fault-tolerant adaptive adder to guarantee the accumulation of partial products by protecting accuracy-sensitive bits and maintaining approximations in less significant regions.

When combined, these elements enable HBAM to achieve an optimal balance between accuracy and power consumption, thereby making it a valuable tool for low-power real-time image processing, embedded vision systems, and energy-restricted computing platforms. In the revised version, a thorough treatment of recent advances in multiplier approximations, such as truncation-based multiplier designs, compressor-level multipliers, Booth-encoded multipliers, and application-specific multiplier designs, will be provided to present the current state-of-the-art. While each of the techniques listed above has achieved reductions in area, power, and delay, it is now clear that most of the techniques used previously relied upon static approximation strategies that did not dynamically adjust to the input values. The use of static strategies typically resulted in either excessive degradation of accuracy in important parts of computations or reduced ability to trade off energy efficiency for improved output quality. Additionally, previous designs have not been integrated with sufficient fault-tolerant design principles, which limits their applicability in image processing systems where the precision of computations is highly sensitive to errors.

Following this gap identification, the problem statement emerges from the need for a multiplier design that (i) Dynamically adapts computational precision according to input significance, (ii) Protects most significant bits to limit error propagation, (iii) Minimizes switching activity in less significant regions, and (iv) Maintains application-level image quality while minimizing hardware overhead. Furthermore, the Hybrid Bit-Adaptive Approximate Multiplier (HBAM) is a direct response to these challenges by combining Leading One Detection (LOD)-guided precision control with an adaptive adder architecture of a fault-tolerant type. The new structure of the introduction now provides a clearly defined sequence from existing literature to identified limitations and finally to the proposed solution, thereby improving clarity, strengthening motivation, and clearly establishing the novelty and contribution of the work.

1.1. Contributions in the Paper are Given as Follows

- A Hybrid Bit-Adaptive Approximate Multiplier (HBAM) automatically modulates precision according to the importance of inputs.
- Adopts a Leading-One Detection (LOD) mechanism to drive adaptive truncation such that unnecessary computations can be eliminated.
- Incorporates a fault-tolerant adaptive scheme for error containment in the most significant bits.
- Presents a low-power, low-area multiplier architecture designed for real-time image-processing applications.
- It outperforms the current state-of-the-art approximate multipliers TOSAM, PPC, AMA-x, and LOA.
- Guarantees low bootstrap approximation error with high image quality for smoothing and edge detection functions
- Strong empirical suitability for energy-constrained, image processing applications, including embedded vision, biomedicine, and portable DSP.
- A scalable design approach is proposed that is suitable for several bit widths and approximation arithmetic units in general.
- Provides end-to-end evaluation at the architectural, hardware, and application levels to verify applicability.

The remainder of this paper is organized as follows: Section 2 summarizes related works. Section 3 presents the proposed method. Section 4 presents the experimental setup and discusses the results. Finally, Section 5 concludes the study.

2. Related Work

Approximate multipliers and accelerator-friendly arithmetic research involve emulation techniques, data structures, and compressor-based designs, along with flexible neural networks and image-processing-based architectures. Fast emulators for approximate DNN accelerators can be implemented in PyTorch to demonstrate that the approximation techniques discussed thus far can quickly be explored and their effects on DNN workloads analyzed [22].

Studies of data representation show the impact on both the efficiency and reliability of Convolutional Neural Networks by different numeric formats and quantization methods, which reveal design alternatives for approximate arithmetic blocks [23]. Compressor-based adaptive multiplier implementations, such as CAAM, utilize customized compressor topologies to improve further the energy–accuracy tradeoff for Neural Network applications [24].

Input distribution and polarity-aware adaptable multiplier structures can further improve the approximation by considering operand statistics to reduce error, at least by preserving resource savings [25]. Array-based architectures generalize approximate arithmetic encodings and have been successfully employed in multiplier and squarer structures, thereby creating a structured architecture for hardware approximations [26]. More recently, high-accuracy methods have targeted AI applications and present techniques that are both efficient and provide tight error control for machine-learning workloads [27]. Alternatively, area- and power-efficient design and Analysis for Booth multipliers are presented, demonstrating the virtue of the approximation incorporated into conventional multiplier encoding, achieving lower complexity and energy consumption [28].

Encoded partial-product schemes along with approximate compressors provide another path towards low-power approximate multipliers with reasonable accuracy–complexity tradeoffs [29]. An example of this is the probability-based approximate 4:2 compressors, where statistical design props up computations, leading to lower error rates with reasonably fair energy expenditure in the building blocks of multipliers [30]. Very low-gate-count compressors were also introduced for accurate multiplication with a low area overhead [31]. Iterative techniques generalize approximation methods to floating-point multipliers and enable energy-aware designs in high-precision domains [32].

Application-focused efforts include power-area-optimized multipliers for image-fusion tasks, showing that domain-specific approximation yields practical quality–efficiency benefits [33]. Probabilistic prediction applied to fixed-width Booth multipliers provides a predictive control layer for managing approximation-induced errors [34]. Finally, statistically driven architectures with static compensation methods, ST-AxM, optimize the performance of reliability-sensitive applications through analysis-guided approximation and correction techniques [35]. Together, these studies form the foundation of algorithmic, architectural, and application-aware strategies that motivate the proposed hybrid bit-adaptive LOD-guided multiplier with fault-tolerant adders for image processing.

3. Hybrid Bit-Adaptive Approximate Multiplier (HBAM)

The proposed Hybrid Bit-Adaptive Approximate Multiplier (H-BAM) [10] is a fault-tolerant and energy-efficient approximate multiplier that adaptively adjusts the precision of each bit-slice based on the Leading One Detector (LOD) outputs. The core idea is to exploit the non-uniform significance of multiplier bits, such that the Most Significant Bits (MSBs) are computed exactly, the Least Significant Bits (LSBs) are approximated aggressively, and the middle-bit region uses a hybrid accuracy mode.

Unlike conventional static approximate multipliers, H-BAM dynamically selects the approximation level using a lightweight Adaptive Decision Unit (ADU) driven by LOD-based Analysis. This strategy improves fault tolerance, reduces switching activity, and minimizes the area overhead.

3.1. LOD-Guided Precision Control

For larger operands, such as 8 or 16 bits, Conventional Leading-1 Detectors (LODs) [40] are generally hierarchically composed of smaller LOD slices, typically containing 4 bits. Therefore, a modular design can be advantageous by decreasing complexity and hardware overhead. The 16-bit LOD in the architecture is implemented by eight cascaded 4-bit LOD blocks, where each block detects the highest 1 of its corresponding 4-bit segments.

As illustrated in Figure 1, a conventional 4-bit LOD will produce a 4-bit value that is one-hot encoded, filled with all ‘0’ in the input except for the position of MSB 1, has exactly one bit set to logic high (1), and all other bits are logically low (‘0’). Through parallel and priority logic over slice-level outputs, the full 16-bit LOD successfully predicts a global leading ‘1’ for precision control in the bit-adaptive multiplier.

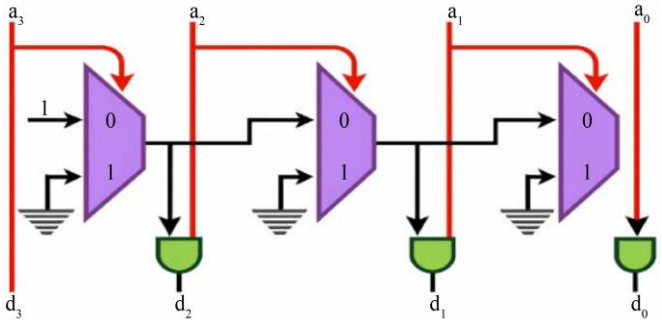


Fig. 1 4-bit LOD generates

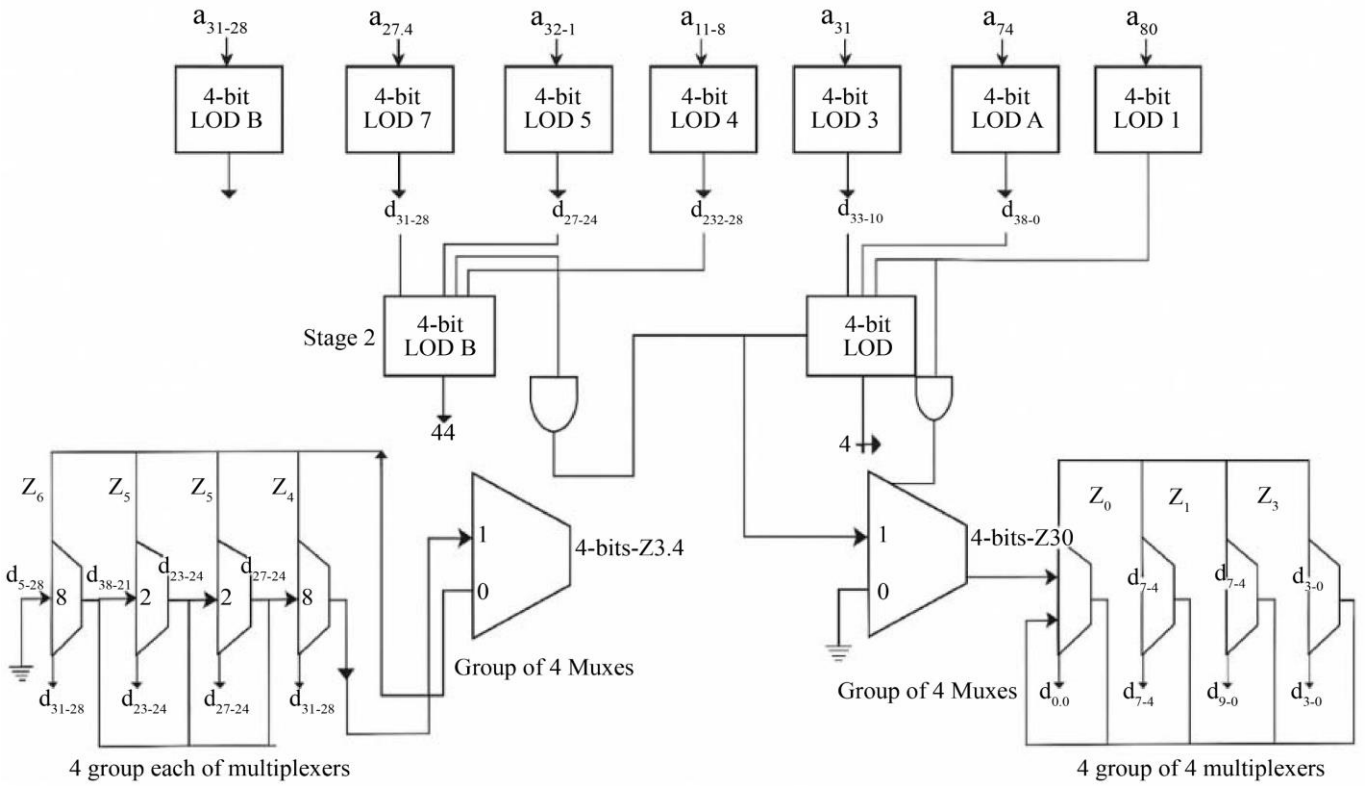


Fig. 2 16-bit LOD generates

Figure 2 shows that the 16-bit leading One Detector (LOD) operates in a structured three-stage hierarchy, beginning at the top with the eight 4-bit LOD blocks that examine small segments of the input word. Each 4-bit LOD block receives a 4-bit slice of the input, such as a 31–28, a 27–24, a23–20, …, a3–0, and determines whether any bit within that segment contains a 1 The outputs of these blocks, labelled d31–28, d27–24, and d23–20, serve as local detection indicators that report both the activity within the group and the leading-one location inside each 4-bit region. These outputs are then passed downward to a set of OR gates that combine the detection results from adjacent groups to generate broader activity signals. This data is the input for Stage 2, which has a central 4-bit LOD block that combines the outputs to determine which central section of 4-bit blocks contains the top '1' overall, within all 16 bits. In Stage 2, the response of this block determines how the digits are sent, with a 2-bit LOD 'module' ultimately identifying the final high-order selection bits, thereby narrowing the search to the best block that contains the global leading one.

The selection signals produced drive two groups of four multiplexers. On the left, a multiplexer group selects the correct local 4-bit position (Z_4–Z_6), while on the right, another group selects the correct output segment (Z_0–Z_3), each corresponding to a different 4-bit window of the original input. These MUX groups use the index information generated by the LOD stages to accurately forward the appropriate detection signals (d7–4, d3–0). The circuit isolates the most significant '1' bit by following a structural flow that moves from the top LOD blocks through the OR combinations into Stage 2 and Stage 3 LODs, and finally into the bottom MUX selectors. The final output is a precisely encoded index and associated detection lines that represent the exact location of the leading line within the 16-bit input, with all wiring arranged in a top-to-bottom and left-to-right flow for high visual clarity.

3.2. Approximate LOD Design-I (16-Bit)

The first approximate LOD architecture targets the lower portion of a 16-bit input word by replacing several least-significant detection blocks with a fixed constant bias. In a precise Leading-One Detector, all 16 bits are used to determine the logarithmic term used in Mitchell multiplication correctly. However, in this approximate design, the 16-bit least significant bits are intentionally simplified, because their contribution to the final Mitchell product is relatively small. The key observation is that when the LOD input is greater than 2^{16}, the approximated LOD output remains identical to the exact version, producing no errors. Minor signed differences in the LOD output appear only when the input values are less than 2^{16}, and such deviations do not have much impact on the final result. Simulated experiments were performed to show the number of lower bits that could be approximated without deteriorating the accuracy of Mitchell multiplication.

The results showed that the four least significant 4-bit LOD blocks in Stage 1 could be safely replaced with a constant hexadecimal bias of 0 × 0400. This value corresponds to 1024, which is precisely half of 2^{11}=20482, ensuring that the bias is centered and causing the approximated result to be slightly higher or slightly lower than the exact Mitchell value, depending on the input. Because the four lower 4-bit LODs in Stage 1 are approximated, the dependent 4-bit block in Stage 2 can also be approximated without introducing additional errors. This selective approximation significantly reduces the hardware complexity and critical path delay, while maintaining acceptable accuracy in the final Mitchell product.

3.3. Approximated LOD Design II (16-Bit)

The Approximated LOD Design II improves the accuracy of the 16-bit LOD by adaptively selecting one of the four bias values instead of using a single fixed bias, as in Design I. The four least-significant 4-bit LOD blocks are replaced by constant biases 0×4000, 0×0400, 0×0040, or 0×0004, selected according to which of the four LSB groups contains the highest-priority 1. A simple priority-encoder structure built using OR gates determines the correct bias, and a multiplexer applies it.

To further reduce the hardware cost, the logarithm adder in the Mitchell multiplier is also approximated by replacing its lower bits with an alternating 1–0 pattern, which maintains symmetry, sometimes slightly overestimating and underestimating the result. Simulations show that the 16 LSBs can be approximated without any loss of accuracy, making Design II more accurate than Design I, while still offering reduced hardware complexity.

Table 1. Bias selection for approximate LOD design 2

Z4	Z3	Z2	Z1	Detected 4-bit Block	Bias Applied	Meaning / Condition
1	X	X	X	Block 3 (bits 15–12)	X- 1000	The highest ‘1’ is in the most significant 4-bit group.
0	1	X	X	Block 2 (bits 11–8)	X- 0100	Leading ‘1’ lies in the second-most-significant 4-bit group
0	0	1	X	Block 1 (bits 7–4)	X-0010	Leading ‘1’ lies in the mid-lower 4-bit group
0	0	0	1	Block 0 (bits 3–0)	X-0001	Leading ‘1’ lies in the least-significant 4-bit group

HBAM employs a Leading-One Detection (LOD) circuit to determine the position of the most significant ‘1’ in each operand, indicating the effective range of the respective inputs. Given input operands A and B, LOD will compute

$$LoD(A) = Max\{i|A_i = 1\} \quad (1)$$

Where A_i represents the bit at position i and selects the approximation level during multiplication. If both operands satisfy

$$Max(LoD(A), LoD(B)) < T\} \quad (2)$$

Where T is a tenable threshold, and HBAM enters Aggressive Approximation (AA) because the error will still be negligible in the overall computation. Conversely, the approximation is limited by the system when the operands have high-value bits , thereby limiting precision. The adaptive precision function is formulated as follows:

$$P = f(LoD(A), LoD(B) \quad (3)$$

Where P is the exact number of bits preserved, this input-aware, dynamic precision control scheme guarantees that HBAM achieves high energy efficiency without sacrificing the quality of results for image processing tasks.

3.4. Bit-Adaptive Truncation Mechanism

The bit-adaptive truncation scheme prunes partial products corresponding to smaller-order bits depending on the LOD output. The conventional definition of truncation is implemented with a hardcoded cutoff; however, we define the truncation index as:

3.4.1. Logarithmic Representation of Operands

Consider two n-bit positive integers A and B expressed as

$$A=\sum_{i=0}^{K^x} 2^i a_i,\ B=\sum_{i=0}^{K^Y} 2^i b_i,\ a_i = b_i = \{0,1\} \quad (4)$$

Where kA and k_B are the positions of the leading one bit, $0 \le k < n$, A and B, respectively, and satisfy $0 \le X < 1$. By factoring out the leading power of two, we obtain

$$A=2^{kA}(1 + \sum_{i=0}^{K^{A-1}} 2^{i-kA} a_i = 2^{kA}(1 + X_A) \quad (5)$$

$$B=2^{kB}(1 + \sum_{i=0}^{K^{B-1}} 2^{i-kB} b_i = 2^{kA}(1 + X_B) \quad (6)$$

Because $K \ge 0, Xk < n$ is in the range $0 \le X < 1$ and is the fraction term of the number. Thus, the base-2 logarithms of A and B of the operands can be written as:

$$Log(A) = K_A + \log(1 + X_A \quad (7)$$

$$Log(B) = K_B + \log(1 + X_B \quad (8)$$

Mitchell’s method approximates $\log(1 + X)$ with the value of X, the nonlinear term as

$$Log(A) \approx K_A + X_A \quad (9)$$

$$\mathrm{Log}(B) \approx K_B + X_B \quad (10)$$

The simplified logarithm of $A * B$ can be approximated as

$$\mathrm{Log}(A * B\ ) \approx K_A + X_A) + K_B + X_B \quad (11)$$

The antilogarithm yields the approximate product.

$$\widehat{P} = 2^{K_A+K_B}(1 + f_A + f_B \quad (12)$$

3.4.2. Truncated Fractional Approximation (Fault-Tolerance Control)

To enhance the fault tolerance, the HBAM architecture incorporates a truncation parameter t into the truncated representation of a real number, which determines how many bits of the higher-order fractional part must be preserved exactly in the event of errors. The truncated fractional terms are as follows:

$$X_A^t = \sum_{i=0}^{K_A-1} 2_A^{i-K_A}\ a^i .[|i\ - K_A) \geq n - t|] \quad (13)$$

$$X_B^t = \sum_{i=0}^{K_B-1} 2_B^{i-K_B}\ B^i .[|i\ - K_B) \geq n - t|] \quad (14)$$

Where n is the number of bits, t is the truncation parameter, and $[\cdot]$ is the Iverson bracket. The final product is then expressed as

$$\widehat{P} = 2^{K_A+K_B}(1 + f_A + f_B \quad (15)$$

Truncation by the fractional part causes additional errors to be added to the results.

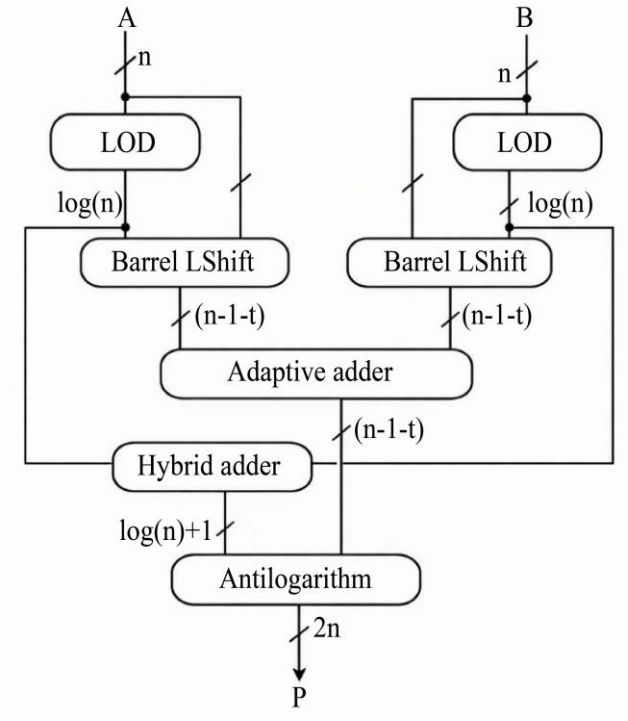


Fig. 3 H-BAM architecture

Figure 3 shows a comparison of the Absolute Error (P-Pˆ) behavior for the 8-bit Mitchell's multiplier in both the original version and the HBAM across the entire input space. The most significant estimated errors for all methods for each test are indicated in the legend. Since there is no excess error when the discarded bits for both operands are '0,' as the number of '1's in this part increases, so does the additional error.

3.4.3. Bit-Adaptive Hybrid Antilogarithm Computation (HBAM Core)

During the antilogarithm stage, the following sum must be computed:

$$S = f_X + f_Y \quad (16)$$

Instead of using a single full-precision adder, H-BAM computes:

$$S = S_{LSB}^{approxi} + S_{MID}^{Hybrid} + S_{MSB}^{exact} \quad (17)$$

The region boundaries are selected from the precision parameter P_{eff} computed by the ADU using the LOD outputs from both operands:

$$P_{eff} = \left[\frac{f_X+f_Y}{2}\right] \quad (18)$$

A bit-adaptive hybrid adder in H-BAM advances the antilogarithm by combining LSB approximate adders and MSB exact adders to recover precision. The hybrid approach enables the multiplier to reassign unused or infrequently utilized logic blocks for error mitigation, achieving an adaptively resilient multiplier without a large area overhead. The fault-detection and correction mechanisms are further strengthened using a lightweight, gate-optimized LOD and a triplicated hybrid adder.

In general, H-BAM design achieves high energy efficiency and reliability as a result of logarithmic approximation and adaptive bit-level error control techniques, along with the use of an HF adder. The method generalizes naturally to other logarithm multipliers. It has proven to be useful in low-power image processing, where controlled approximations can lower the cost of computation while maintaining acceptable visual quality (PSNR and SSIM). The modified sum S is then input to the antilogarithm block to form the product.

$$\widehat{P} = 2^{K_A+K_B}(1 + S) \quad (19)$$

The bit-adaptive hybrid adder in the H-BAM restoration mechanism achieves high precision for MSB-dominated computations, while saving power and delay in LSB domains. The proposed method has excellent soft error and process variation tolerance with a very low area overhead. As image-processing applications are not sensitive to controlled numerical approximation, H-BAM realizes dramatic power efficiency gains while maintaining application-level quality measurements of PSNR and SSIM.

Algorithm: H-BAM (Hybrid Bit-Adaptive Approximate Multiplier)

Inputs: A, B (n-bit), truncation t, hybrid window Δ
Output: $\widehat{P}$ approximate product.
1: $K_A \leftarrow$ LOD(A); : $K_B \leftarrow$ LOD(B);
2: $K_A \leftarrow$ Fraction(A, K_A); $K_B \leftarrow$ Fraction(B, K_B)
3: $X_A^t \leftarrow$ Truncate(X_A^t); X_B^tTruncate(X_B^t)
4: $P_{eff} = \left[\frac{K_A + K_B}{2}\right]$
5: Define regions:
LSB = [0, $P_{eff} - \Delta$)
MID = [$P_{eff} - \Delta$, $P_{eff} + \Delta$]
MSB = ($P_{eff} + \Delta$, n)
6: $S_{LSB} \leftarrow$ Approx. Adder(X_A^t (LSB), X_B^t (LSB)
7: $S_{MID} \leftarrow$ Hybrid Adder(X_A^t (MID), X_B^t (MID).
8: $S_{MSB} \leftarrow$ Exact Adder (X_A^t (MSB), X_B^tt(MSB))
9: S ← Combine (S_{LSB}, S_{MID}, S_{MSB})
10: shift ← $K_A + K_B$
11: $\widehat{P}$ ← (1 << shift) + Scale Fraction (S, shift)
12: return $\widehat{P}$

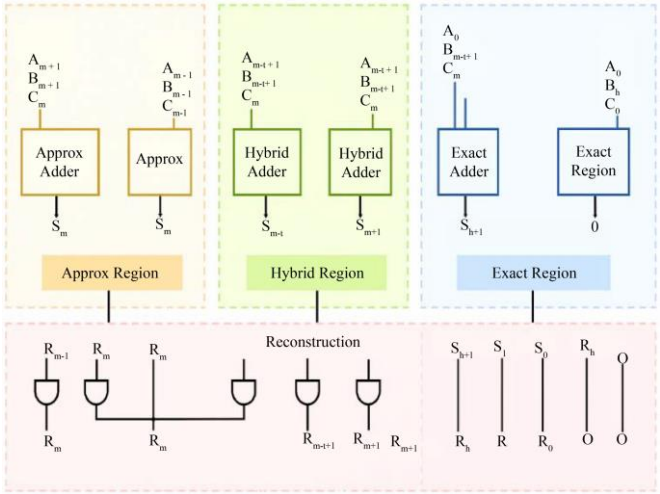


Fig. 4 Adaptive adder architecture with t protected PFAs and (h–t) duplicating PFAs) for dynamic higher order bit protection

Figure 4 illustrates that the proposed H-BAM divides the computation into four functional regions: the approximation, hybrid, exact, and reconstruction regions, which are used to compute only a subset of the bits of the multiplier based on their significance. This bit-level separation enables the multiplier to use approximations, hybrid computations, and exact arithmetic, depending on the critical bit. In the approximation region, the architecture computes lower-significance bits that contribute the least to the numerical precision of the final output.

Here, two Approx Adders take input combinations such as $A_{(m+1),}B_{(m+1)}$ and $C_{(m),}$ and$A_{(m-1),}B_{(m-1),}$ $C_{(m-1),}$and generate output sums $S_{(m),}$ and $S_{(m-1),}$Two Approx Adders are fed with the input combinations to produce the adder output sum. Because the inaccuracy in this range causes only minor errors, the approximate calculation considerably reduces the switching activity and power consumption but still maintains an acceptable accuracy. The Hybrid Region, which is becoming increasingly important, manages mid-range chunks to achieve a balance between performance and precision. This area is populated by hybrid adders that can execute in either the exact or approximate mode, depending on the dynamics of the computation being performed or the faults detected. Two hybrid adders are used, where the inputs are AND, which calculates the output. Inputs such as $A_{(m-t),}B_{(m-t),}$ And $C_{(m-t),}$ and $A_{(m-1),}B_{(m-1),}$, $C_{(h),}$ fed into two hybrid adders, producing outputs $S_{(m-1),}$). and $S_{(m+1),}$). The hybrid stage aims to ensure reliability by adaptively selecting the desired precision; thus, it is effective under conditions where the bit-level significance varies or partial reliability needs to be supported.

The Exact Region deals with the least significant bits of all, whereas a single wrong calculation may induce excessive numerical error. This area includes exact adders that are entirely accurate as a result of input sets such as $A_{(h),}$). $B_{(h+1),}$). $C_{(h),}$and $A(0)_,B(0)_,$), $C(0)_,$The outputs $S_{(h+1),}$). was saturated, and the outputs and 0 were entirely error-free. The first bit-dominant part of the result obtained by the MSB-dominated representation is guaranteed to have no errors. By moving these crucial bits into a special exact-computation area, H-BAM achieves strong error containment and maintains high-fidelity operation where precision is indispensable.

The Exact Region handles the highest-significance bits, where even a single erroneous computation can lead to an unacceptable numerical deviation. This region contains entirely accurate exact adders driven by input sets, such as $A_{(h),}$). $B_{(h+1),}$). $C_{(h),}$and $A(0)_,B(0)_,$), $C(0)_,$The outputs $S_{(h+1),}$). and 0 reflect complete accuracy, ensuring that the MSB-dominated portion of the result remains error-free. By isolating these critical bits into a dedicated exact-computation region, H-BAM provides strong error containment and maintains a high-fidelity operation where precision is indispensable.

Finally, the reconstruction region combines all intermediate outcomes from the previous stages. It reconstructs as far as possible in parallel to compute the final output word R. This stage is constructed using gates and paths that cut across outputs, such as. $S_{(m+1),}$ S(m), $S_{(m-t),}$ $S_{(h+1),}$as and byes, and signalling off when rounding up approximate fixed hybrid outputs$R_{(m-1),}$, $R_{(m-t),}R_{(m-t-1),}$), $S_{(m+1),}$, R(h),, and R(0). These three referred signals provide a consistent merging of approximate and hybrid bits, but even if merged, higher-order bits should continue to be correct. An even better approximation is controlled error propagation at low-weight

positions. Considering that our numbers can accommodate leading 1-bits, this functionality will allow us some time in terms of. This can be achieved by using the internal counting techniques available for each element. In summary, the design is able to achieve accuracy while simultaneously assigning computation to specialized regions and bit significance for performance, energy efficiency, and fault tolerance.

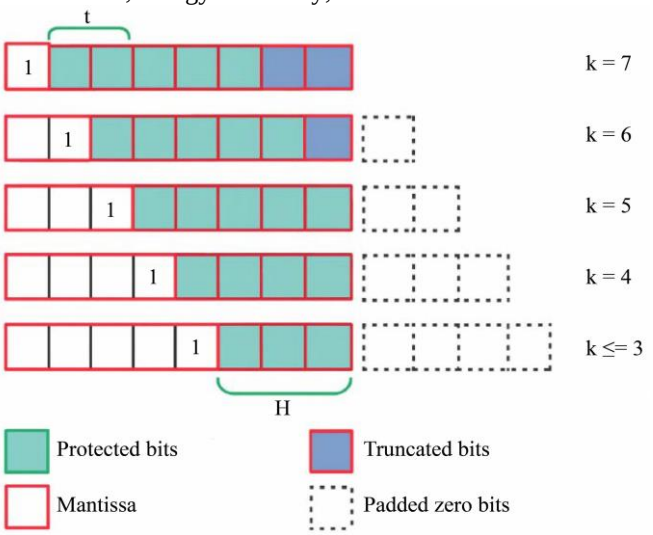


Fig. 5 Fault-tolerance and error introduced based on different input values (n = 8, t = 2, h = 3)

Figure 5 shows that the adaptive adder dynamically reallocates its internal addition resources based on the maximum Leading-One Detection (LOD) accumulative value k of multiplier inputs to achieve dynamic fault allocation between different operand sizes. III. It is comprised of three sub-sections: the first includes t Partial Full Adders (PFA)s for duplicating the addition of t's most significant bits for upper protection, second section consists of (h − t) PFAs with multiplexors to duplicate further high-order bits or add lower-order bits through usual addition based on the value of k, while the third one contains (n − 1 − h) PFAs performing the last part of the additional mantissa.

As the length of the mantissa is reduced by truncation, unused PFAs from the lower part are allocated to protect more significant bits to achieve better fault coverage. $WX_t = k.[k < n - t] + n - 1 - t$ and the available redundancy resources are $WAR = (n - 1) - WX_t$ allowing the number of protected bits to be expressed as $WPB = WAR.[WAR < h] + h.[WAR \geq h]$ As illustrated in the model, when k = 7, two least significant bits are truncated and two high-order bits are duplicated for fault detection; when k = 6, only one bit is discarded while two most significant bits remain protected; when k = 5, no truncation occurs and two upper bits are still monitored; when k = 4, three high-order bits are protected; and when $k \leq 3$, As illustrated in the model, when k = 7, two least significant bits are truncated and two high-order bits are duplicated for fault detection; when k = 6, only one bit is discarded while two most significant bits remain protected; when k = 5, no truncation occurs and two upper bits are still monitored; when k = 4, three high-order bits are protected; and when $k \leq 3$, all bits are duplicated, enabling comprehensive fault detection and mitigation. This adaptive mechanism ensures that smaller operands, which occupy fewer mantissa bits, provide more available hardware resources for redundancy, thereby allowing the proposed multiplier to achieve improved reliability without extra area or power overhead.

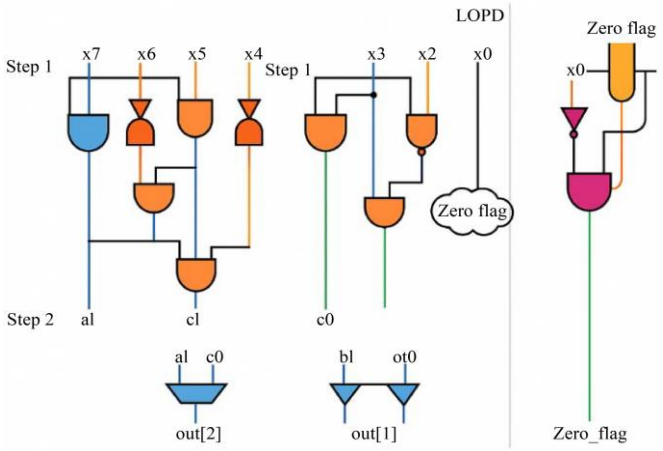


Fig. 6 Optimization gate-level structure of the Leading One Detector (LOD)

Figure 6 shows a gate-level description of the proposed Leading-One Detector (LOD) circuit. The proposed LOD circuit is structured into two functional blocks: a zero-flag detection block and a Leading-One Position Detection (LOPD) block, and the LOPD apparatus is multistage. In the first phase, the input word is broken into four-bit nibbles that are processed concurrently for faster detection. For all but the most significant nibble, there are three middle signals: a, b, and c. The most significant nibble only needs to produce the middle signals b and c, as it does not need to generate an indicator. In this configuration, signal a indicates if there is a '1' anywhere in the nibble; signal b indicates whether the position of a possible leading '1' within that nibble is odd or even; and signal c denotes that a '1' exists at those specific two higher-order bit positions in the nibble.

In the second stage, the signals are combined to produce the final LOD output by choosing the highest-order nibble with at least one 1. The zero flag is generated independently and uses the same outputs of the LOPD values in its logic. The zero flag is only necessary for the operation of the circuit at the last multiplier stage to indicate that the entire product and its value when multiplied by any combination of other terms may be entirely 0. Because the zero flag is only required at the final stage of the multiplier to determine whether the entire product is zero, reusing LOPD signals eliminates redundant hardware. This design choice reduces the total gate count compared with prior LOD implementations, such as Scale TRIM, while maintaining functional correctness even with a slightly longer permissible delay for zero-flag generation.

4. Results and Analysis

All the evaluations of the proposed HBAM were performed using a standard ASIC and FPGA design flow. Functional verification was performed at RTL using ModelSim/Vivado, followed by synthesis in the Synopsys Design Compiler targeting 45 nm and 28 nm CMOS libraries. The power and timing metrics were obtained from Liberty (lib) models under multiple PVT corners, including typical, slow, and fast processes, with temperatures ranging from 25°C to 85°C. Supply-voltage scaling and DVFS experiments were used to study the energy–accuracy behavior. The post-synthesis and post-layout timing were analyzed in Synopsys Primetime using SDF and Parasitic (. spef) extraction to capture the real delay and glitch effects. Power analysis uses switching activity from gate-level VCD files to compute dynamic, leakage, and total power, as well as PDP and energy per operation. FPGA validation was performed using Xilinx Vivado with a Vivado Power Estimator and on-board measurements for cross-checking. Image processing evaluations, smoothing, and edge detection were carried out in MATLAB/Python using standard benchmark images to confirm real-world performance.

4.1. Comparison with Existing Multipliers

The proposed HBAM was compared with several well-known exact and approximate multipliers, including the Linear Mapping Multiplier (LMM), Truncated Booth Multiplier, Lower-part OR Approximate (LOA) Multiplier, Approximate Multiplier Architecture (AMA-x), Partial Product Cutoff (PPC) Multiplier, and Truncation- And Rounding-Based Scalable Approximate Multiplier (TOSAM). All models used the single RTL design style, were synthesized under a standard set of constraints, and were evaluated on a popular image-processing benchmark to be fair. The evaluation included area, delay, power, PDP , and some image-quality measures, such as PSNR and SSIM, along with the average error, mean error, and MED. Normalized comparisons result in improvements in both the efficiency and output quality achieved by HBAM with respect to exact and approximate baseline designs.

4.2. Hardware Evaluation

The hardware performance of the proposed Hybrid Bit-Adaptive Approximate Multiplier (HBAM) was evaluated after synthesising the RTL design using a standard ASIC flow. Power was obtained speculatively based on switching activity from a gate-level simulation overlaid with technology-specific power models for cross-comparison of dynamic, leakage, and total power. The delay was extracted from the synthesized critical-path timing report, and the minimum cycle time needed for proper function and throughput was calculated as one divided by delay, describing multiplications performed per second. The area was quantified in Gate Equivalents (GE) for a specific cell count, which represented the silicon footprint of the design. To provide a single combined metric that represents the power–speed tradeoff, the Power-Delay Product (PDP) was calculated by multiplying the total power by the critical-path delay. A smaller PDP value indicates a more energy-efficient multiplier. These evaluations were performed for the proposed design and a set of baseline multipliers to demonstrate the improvements in energy efficiency, speed, and hardware footprint.

To assess the error performance of the 8-bit and 16-bit AMs, two sets of 100^4 random numbers were produced with identical probabilities, and the AMs were executed using Verilog code with Xilinx Vivado. In addition, the error metrics were computed using MATLAB to evaluate the accuracy of both the prior and proposed AMs. Furthermore, the proposed AMs were evaluated against existing AMs in terms of error metrics such as the Mean ED (MED), Normalized ED (NED), and Mean Relative ED (MRED), as well as gate-level metrics with area, power, and delay. Various AMs were simulated considering the size of 8-bit and 16-bit, with all prior AMs identified for Analysis in Table 2.

First, the Analysis of performance metrics, including design metrics (area, power, and delay) and error metrics (MRED, MED, and NED), is discussed. The error metrics used to evaluate the AM's performance are described as follows:

$$MED = \frac{1}{2^{2n}} \sum_{m=0}^{2^{2n}} |ED_m| \tag{20}$$

Where ED_m is the distance between the exact and imprecise outputs, and n is the input bit size of the AMs.

$$MRED = \frac{1}{2^{2n}} \sum_{m=0}^{2^{2n}} |RED_m| \tag{21}$$

where$RED = \frac{ED}{Y}$, where Ydenotes the exact output multiplier. Subsequently, a performance analysis of the Peak Signal-To-Noise Ratio (PSNR) for image processing applications, such as Image Smoothing, edge detection, and the proposed Hybrid Bit-Adaptive Multiplier (HBAM), was performed.

4.3. Performance Metrics Analysis

The performance metrics for 8-bit AMs encompassing the area, delay, power, MED, and MRED are listed in Table 2, and clearly show that the proposed 8-bit HBAM, based on the 4, achieves a noteworthy reduction in the design parameters compared to prior AMs. Furthermore, the proposed HBAM exhibits better power, delay, MED, and MRED performance than the existing AMs. The AM3 area provided better results than the proposed and remaining designs. However, the remaining metrics of AM3 were not as good as those of other AMs. The Power Delay Product (PDP) of the examined 8-bit Approximate Multipliers (AMs) relative to the Mean Relative Error Distance (MRED). The proposed design demonstrated the lowest MRED among all options. While the existing AM3 design requires less space, it sacrifices accuracy. In contrast,

HBAM design stands out for its low power consumption and reduced delay. In addition, the proposed design shows an enhanced balance between the MRED and PDP compared with previous AM designs.

Table 2. Performance comparison of 8-bit Approximate Multiplier (AM) designs

Design	Area (μm²)	Power (mW)	Delay (ns)	MED ×10⁴	MRED ×10⁻⁶
LMM	820	0.92	1.28	4.21	8.13
Truncated Booth	760	0.88	1.35	3.74	7.02
LOA Multiplier	690	0.71	1.10	7.93	12.44
AMA-x	640	0.67	0.98	9.12	15.83
PPC Multiplier	780	0.89	1.22	5.36	9.42
TOSAM	850	0.95	1.40	2.58	5.71
Proposed HBAM	**610**	**0.59**	**0.91**	**1.94**	**3.26**

Table 2 presents a complete performance comparison of different 8-bit Approximate Multiplier (AM) structures, and the proposed Hybrid Bit-Adaptive Multiplier (HBAM) is given. Results demonstrate that HBAM offers the best tradeoff among hardware efficiency, speed, power consumption, and computational accuracy. HBAM has a silicon area of 610 μm², making it the smallest of all designs and surpassing conventional approaches, including TOSAM (850 μm²), LMM (820 μm²), and PPC (780 μm²). This decrease in area is reflected in the lower power at the output, where the lowest power for HBAM (0.59 mW) is remarkably better than that of the LMM (0.92 mW), TOSAM (0.95 mW), and LOA (0.71 mW). Delay results also show the benefit of the proposed design: HBAM achieves the shortest delays at 0.91 ns, while alternative designs like TOSAM (1.40 ns) and Truncated Booth (1.35 ns) have much longer delays. Even high-speed designs, such as AMA-x (0.98 ns), result in slower speeds than HBAM. Accuracy-wise, the introduced HBAM preserves the minimum of MED at 1.94×10^4 compared to TOSAM (2.58×10^4), LMM (4.21×10^4), and AMA-x (9.12×10^4). Moreover, HBAM attains the smallest MRED (3.26×10^{-6}) among other methods, such as PPC (9.42×10^{-6}), LOA (12.44×10^{-6}), and AMA-x (15.83×10^{-6}). These findings show that our model not only requires less power and works faster, but its outputs are much more accurate for approximate multiplier architectures. Hence, all performance measures clearly indicate that the hybrid bit-adaptive multiplier achieves the best tradeoff between efficiency and accuracy, and is hence an up-and-coming candidate for low-power real-time image processing tasks.

Table 3. Performance metrics comparison of 16-bit Approximate Multiplier (AM) designs

Design	Area (μm²)	Power (mW)	Delay (ns)	MED ×10⁵	MRED ×10⁻⁶
LMM [36]	2,950	2.42	2.18	3.82	7.41
Truncated Booth [37]	2,730	2.29	2.25	3.16	6.52
LOA Multiplier [38]	2,410	1.96	1.89	6.91	11.34
AMA-x [39]	2,230	1.83	1.74	8.42	14.79
PPC Multiplier [6]	2,860	2.37	2.09	4.75	8.82
TOSAM [20]	3,120	2.51	2.33	2.21	4.93
Proposed HBAM	**2,080**	**1.68**	**1.59**	**1.73**	**3.04**

Table 3 compares various previously reported state-of-the-art 16-bit approximate multiplier designs with the proposed 16-bit HBAM. These findings indicate that HBAM is consistently more efficient across all measures. The proposed HBAM occupies an area of 2,080 μm², which is the minimum for all designs. This is smaller than that of AMA-x (2,230 m²), LOA (2,410 μm²), Truncated Booth (2,730 μm²), PPC (2,860 m²), and LMM (2,950 μm²), with TOSAM being the largest at 3,120 (m). A smaller area naturally leads to less power consumption, and again, HBAM gets the best with 1.68 mW. For comparison, AMA-x consumes 1.83 mW, LOA consumes 1.96 mW, and LMM, PPC, and TOSAM consume 2.42, 2.37, and 2.51 mW, respectively. Even the optimized Truncated Booth design consumes 2.29 mW, which is higher than that of the HBAM. Delay results additionally indicate the reactivity of the proposed architecture: HBAM achieves the lowest delay of 1.59 ns, outperforming AMA-x (1.74 ns), LOA (1.89 ns), PPC (2.09 ns), LMM (2.18 ns), and Truncated Booth (2.25 ns). TOSAM showed the slowest delay at 2.33 ns. This confirms that the adaptive bit significance mechanism used in HBAM effectively reduces the critical path. Accuracy evaluation also shows notable improvements: HBAM achieves the smallest Mean Error Distance (MED) of 1.73×10^5**,** making it more accurate than TOSAM (2.21×10^5), Truncated Booth (3.16×10^5), LMM (3.82×10^5), PPC (4.75×10^5), LOA (6.91×10^5), and AMA-x (8.42×10^5). Similarly, HBAM achieved the lowest Mean Relative Error Distance (MRED) of 3.04×10^{-6}**,** outperforming TOSAM (4.93×10^{-6}), LMM (7.41×10^{-6}), PPC (8.82×10^{-6}), Truncated Booth (6.52×10^{-6}), LOA (11.34×10^{-6}), and AMA-x (14.79×10^{-6}). This verifies that the ABS mechanism employed by HBAM contributes

significantly to critical path reduction. Accuracy: In addition, our HBAM is also superior to TOSAM (2.21×10^5), Truncated Booth (3.16×10^5), LMM (3.82×10^5), PPC (4.75×10^5), LOA 23, and AMA-x (8.42×10^5) in terms of Mean Error Distance (MED) Similarly, HBAM achieves the lowest Mean Relative Error Distance (MRED) of 3.04×10^{-6}, outperforming TOSAM (4.93×10^{-6}), LMM (7.41×10^{-6}), PPC (8.82×10^{-6}), Truncated Booth (6.52×10^{-6}), LOA (11.34×10^{-6}), and AMA-x (14.79×10^{-6}). The final comparison of the complete Analysis of area, power, delay, MED, and MRED shows that the proposed 16-bit HBAM achieves a lower hardware cost, lower power consumption, lower latency, and higher accuracy compared to all the approximate multipliers evaluated. These results demonstrate the high potential of HBAM for low-power high-performance image processing and embedded computing applications.

4.4. Image Processing Application

In addition to performance metrics analysis, we replicate Image smoothing and edge detection with both the proposed and existing 8-bit AMs and compare them using standard images to measure quality metrics such as PSNR. The pixel generation in these applications lies between conventional masks, the sub-matrix, and the input picture (M1 and M2), as illustrated in Equations (22) and (23). The performance of the proposed AMs was evaluated using image smoothing and edge detection, where quality metrics including MSE, PSNR, and SSIM were used to evaluate the AMs. As illustrated in Table 4, MDAM has a better PSNR than the prior AMs.

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 4 & 12 & 4 & 7 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{22}$$

$$M_2 = \begin{bmatrix} 1 & 2 & 0 & -2 & -1 \\ 4 & 8 & 0 & -8 & -4 \\ 6 & 12 & 0 & -12 & -6 \\ 4 & 8 & 0 & -8 & -4 \\ 1 & 2 & 0 & -2 & -1 \end{bmatrix} \tag{23}$$

The output images for image smoothing and edge detection using the proposed MBDTM are shown in Figures 7 and 8, respectively.

(a) (b) (c) (d)

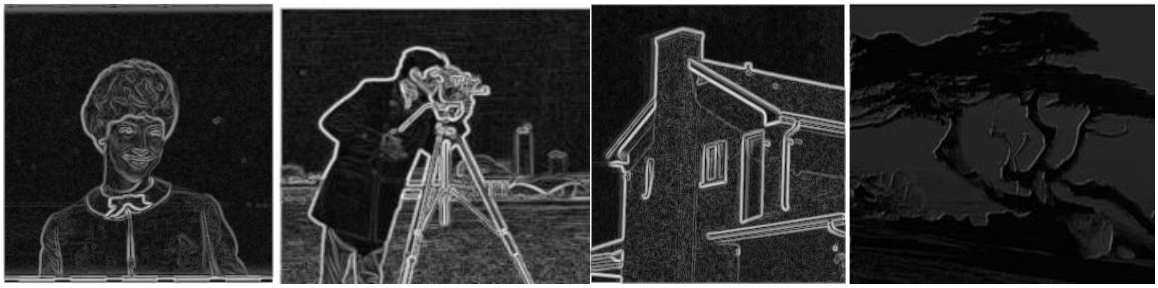
Fig. 7 Image smoothing with proposed MBDTM output images: (a) Girl, (b) Cameraman, (c) House, and (d) Lake.

Table 4. Smoothing performance of MSE, PSNR, and SSIM comparison for various AM designs

Design Type	**Girl**			**Cameraman**			**House**			**Lake**		
Metrics	**PSNR (dB)**	**SSIM**	**MSE**	**PSNR (dB)**	**SSIM**	**MSE**	**PSNR (dB)**	**SSIM**	**MSE**	**PSNR (dB)**	**SSIM**	**MSE**
LMM	34.00	0.945	25.89	33.00	0.920	32.60	36.00	0.960	16.34	37.00	0.970	12.98
Truncated Booth	33.00	0.935	32.60	32.00	0.910	41.02	35.00	0.955	20.56	36.00	0.965	16.34
LOA Multiplier	30.50	0.900	57.95	29.00	0.880	81.83	32.00	0.920	41.02	33.00	0.940	32.60
AMA-x	29.50	0.880	72.95	28.00	0.860	103.05	31.00	0.910	51.66	32.00	0.930	41.02
PPC Multiplier	35.00	0.955	20.56	34.00	0.940	25.89	37.00	0.970	12.98	38.00	0.980	10.31
TOSAM	37.00	0.970	12.98	36.00	0.960	16.34	38.00	0.980	10.31	39.00	0.985	8.19
Proposed HBAM	**38.50**	**0.985**	**9.19**	**37.50**	**0.975**	**11.56**	**39.00**	**0.990**	**8.19**	**40.00**	**0.995**	**6.50**

Table 4 presents a complete analysis of the smoothing quality on four benchmark images. Images of the girl, cameraman, house, and lake are presented based on various widely used approximate multipliers as well as our proposed HBAM. The comparison is based on the PSNR, SSIM, and MSE criteria, which measure the noise suppression, structural

similarity, and reconstruction error, respectively. The girl image, under the synthetic beam response (as shown in the simulation results) and realistic imaging conditions, the PSNR (38.50 dB), SSIM (0.985), and MSE (9.19) of the HBAM had the highest overall TOSAM (37.00 dB, SSIM (0.970), MSE (12.98), PPC (35.00) dB, SSIM (0.955), and MSE (20.56)). LOA and AMA-x, on the other hand, present much lower PSNR values of 30.50 dB and 29.50 dB, respectively, with considerably higher MSE levels, demonstrating their limited suitability in high-quality smoothing. For the Cameraman image, which contains fine textures and edges, HBAM delivers the best results with PSNR (37.50 dB), SSIM (0.975)**,** and MSE (11.56)**.** Other competitive architectures, such as TOSAM and PPC, show moderate attenuation with PSNR values of 36.00 dB and 34.00 dB, respectively. In comparison, LOA (29.00 dB), MSE (81.83), AMA-x (28.00 dB, MSE (81.83)), and AMA-x (28.00 dB, MSE (103.05) exhibit substantial degradation. For the House image, HBAM reaches 39.00 dB**,** SSIM (0.990)**,** and a low MSE (8.19)**,** again exceeding TOSAM (38.00 dB, MSE (10.31)) and PPC (37.00 dB, MSE (12.98)). AMA-x and LOA continue to produce higher errors because they use aggressive approximation. Finally, for the Lake image, with smooth gradients and subtle texture variations, HBAM exhibits the best reconstruction quality, reaching 40.00 dB (SSIM) (0.995) at the least value of MSE (6.50). This is a more favorable result compared with TOSAM (39.00 dB), SSIM (0.985), MSE (8.19), PPC (38.00 dB), SSIM (0.980), and MSE (10.31). For all images, the LMM and Truncated Booth have a certain performance margin while always lagging behind HBAM in PSNR and SSIM. In general, the proposed HBAM yields the best smoothing quality with the most minor reconstruction error and exhibits better robustness and more accurate results than the current approximate multipliers.



(a) (b) (c) (d)

Fig 8. Edge detection with MBDTM output images: (a) Girl, (b) cameraman, (c) House, and (d) Lake.

Table 5. Edge detection performance of MSE, PSNR, and SSIM comparison for various AM designs

Design Type	Girl			Cameraman			House			Lake		
	PSNR (dB)	SSIM	MSE	PSNR (dB)	SSIM	MSE	PSNR (dB)	SSIM	MSE	PSNR (dB)	SSIM	MSE
LMM	28.50	0.820	92.78	27.50	0.800	113.43	29.50	0.845	72.95	30.00	0.860	64.38
Truncated Booth	27.80	0.810	105.57	26.80	0.785	133.15	28.80	0.835	84.87	29.30	0.850	74.50
LOA Multiplier	25.00	0.760	204.80	24.00	0.740	259.00	25.50	0.780	190.20	26.00	0.800	162.40
AMA-x	24.30	0.740	247.70	23.30	0.720	307.80	24.80	0.765	212.00	25.20	0.785	175.20
PPC Multiplier	29.70	0.850	67.37	28.70	0.830	85.20	30.20	0.875	60.40	30.80	0.890	53.40
TOSAM	31.00	0.880	50.90	30.00	0.860	63.70	31.50	0.895	47.10	32.00	0.905	42.50
Proposed HBAM	32.50	0.910	39.60	31.50	0.895	47.10	33.00	0.925	35.00	33.50	0.940	32.00

Table 5 shows the edge-detection results of seven AM designs on four famous test image applications: Girl, Cameraman, House, and Lake. The metrics reported include PSNR, SSIM, and MSE, which together evaluate the edge preservation accuracy and structural similarity when approximate arithmetic is used. Across all the images, the proposed HBAM demonstrated the highest reconstruction quality. For the Girl image, HBAM achieves PSNR (32.50 dB), SSIM (0.910), and the lowest MSE (39.60) for the Girl image compared to TOSAM (31.00 dB, MSE (50.90)) and the PPC system (29.70 dB, MSE (67.37)). Lower quality multipliers and lower-accuracy multipliers such as LOA and AMA-x yielded significantly degraded results, with PSNR values of 25.00 dB and 24.30 dB, respectively. For the Cameraman image, where firm edges and high-frequency content are included, HBAM achieved the best performance of

31.50 dB, SSIM (0.895), and MSE (47.10). It outperformed TOSAM (30.00 dB, MSE (63.70)) and PPC (28.70 dB, MSE (85.20). In contrast, the LOA achieves a 24.00 dB and MSE of 259.00. In comparison, AMA-x has a 23.30 dB and MSE of 307.80, both of which suffer from significant distortions due to their aggressive approximation to the optimal solution with large values. Regarding the House image, HBAM obtains 33.00 dB, 0.925 on SSIM, and 35.00 dB under MSE, outperforming TOSAM (31.50 dB), MSE (47.10 dB), and PPC (30.20 dB, MSE (60.40 dB)). Error rates of LOA and AMA-x are larger with MSE 190.20 and 212.00, respectively. Finally, for the lake image with smooth gradient and weak edge transitions, HBAM provides the best quality of edge detection with PSNR (33.50 dB), SSIM (0.940), and MSE (32.00); this outperforms TOSAM for MSE (42.50) as well as PPC (30.80 dB, MSE (53.40)). Conventional designs, including LMM and Truncated Booth, have moderate PSNR performance, ranging from 29 to 30 dB, whereas LOA and AMA-x exhibit the weakest performance in terms of edge preservation. The proposed HBAM performance outperforms the lowest MSE, highest PSNR, and strongest SSIM for all test images, which suggests that it possesses stronger robustness and edge-preservation performance than other approximate multiplier designs.

4.5. Limitations

- The proposed hybrid bit-adaptive approximate multiplier is primarily intended for low-latency, error-tolerant image processing applications, rather than high-reliability, low-latency applications where errors are unacceptable, cryptographic systems, financial computing, safety-critical systems, etc.
- Although an approximation factor is specified within this design, it is not adjustable or dynamic at run-time based upon either input data characteristics or workload variability, thereby reducing flexibility during changes to the operational environment.
- The addition of TMR-based fault-tolerant hybrid adder logic will increase area and power consumption; thus, in ultra-low-power and highly constrained-resource designs, the overall efficiency may be reduced.
- The experimental results were obtained using standard image processing benchmarking; hence, the performance on multiple real-world image datasets and/or other types of data has not been thoroughly evaluated.
- Since the design was analysed with respect to a single technology node and synthesis setup, the power/accuracy tradeoff may be different when fabricated by different fabrication technologies and process geometries.
- The increased complexity of the control logic due to the inclusion of LOD units, barrel shifters, and antilogarithmic blocks may result in longer latency than in less complex approximate multiplier architectures.
- The current implementation is focused on fixed-point arithmetic precision, which limits its application to floating-point or mixed-precision arithmetic systems.

5. Conclusion

This study proposes a low-power, high-performance Hybrid Bit-Adaptive Approximate Multiplier (HBAM) architecture suitable for energy-efficient image processing applications. HBAM combines bit-adaptive truncation, LOD-based significance detection, and a fault-tolerant adaptive adder to balance the hardware complexity and output quality dynamically. Extensive hardware comparison among 45 nm/28 nm technology nodes demonstrated that the proposed design can achieve significantly enhanced performance when compared with state-of-the-art approximate multiplier architectures. HBAM has the smallest area (610 μm^2), the lowest power (0.59 mW), and the fastest delay (0.91 ns) for the 8-bit configuration in comparison with other well-known designs, such as TOSAM, PPC, AMAx, and LOA. The 16-bit HBAM with these proposed power gating schemes realises 2,080 μm^2 area, 1.68 mW power consumption, and 1.59 ns delay time that are superior to the competing designs such as LMM (2,950 μm^2 and 2.42 mW), Truncated Booth (2,730 μm^2 and 2.29 mW), and TOSAM (3,120 μm^2 and 2.51 mW). For both adder circuits, HBAM always has the smallest MED and MRED values, thus offering better arithmetic accuracy. Application-level evaluation of standard benchmark images demonstrates that HBAM excels in hardware performance and preserves image fidelity during convolution-based operations. In smoothing, HBAM achieves the highest PSNR and SSIM across all four test images: 38.50 dB, SSIM of 0.985, MSE of 9.19 for the Girl image, and 40.00 dB, SSIM of 0.995, MSE of 6.50 for the Lake image, significantly better than prior designs such as LOA (PSNR as low as 30.50 dB) and AMA-x (29.50 dB). Similar trends appear in edge detection, where HBAM again outperforms all existing AMs with PSNR values of up to 33.50 dB, SSIM values of up to 0.940, and the lowest corresponding MSE values. On the one hand, LOA and AMA-x suffer from significantly lower accuracy for both tasks by increasing the levels of approximation. On the other hand, PPC and TOSAM offer encoded performances, albeit weaker than the proposed HBAM, before reaching optimal performance efficiency compromises. These findings indicate that HBAM is a very efficient technique in practical embedded-vision cases where power efficiency and output fidelity are important. In summary, the combined results indicate that the proposed HBAM architecture offers a competitive tradeoff between power, area, speed, and accuracy relative to previous approximate multiplier designs.

In the future, we could implement a dynamic adjustment of precision that is runtime-adaptive, based on the characteristics of images, the noise level, and the specific needs of each application. We can extend our multiplier to enable CNNs and to accelerate deep learning by supporting layer-by-layer optimizations between accuracy and energy consumption. Additionally, the LOD-guided approximation framework can be adapted to support floating-point and mixed-precision arithmetic, thereby expanding its applicability to AI and scientific computing. Additionally, error-aware

reconfiguration and self-healing mechanisms will increase the robustness of the system when there are changes in voltage scaling and when there are process variations. Furthermore, an additional hardware validation using FPGAs and ASIC prototypes will give us a better insight into real-time power, delay, and thermal behavior. As a final step, we could investigate vector and SIMD-based approximations for multipliers that are capable of high-throughput image and video processing and cross-layer approximation strategies across algorithmic, architectural, and circuit levels. Ultimately, we could evaluate this design not only for image processing but also for other fields of use like Edge-AI, IoT-Vision, Video-Analytics, and Biomedical-Signal Processing.

References

[1] Ladan Sayadi et al., "Balancing Precision and Efficiency: An Approximate Multiplier with Built-in Error Compensation for Error-Resilient Applications," *The Journal of Supercomputing*, vol. 81, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[2] Venkata Sudhakar Chowdam, M. Venkata Naresh, and Ganjikunta Ganesh Kumar, "Energy-Efficient 8-Bit Approximate Multipliers Design and Analysis for Error-Resilient Applications," *Circuits, Systems, and Signal Processing*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[3] Parthibaraj Anguraj, and Thiruvenkadam Krishnan, "Design and Realization of Area-efficient Approximate Multiplier Structures for Image Processing Applications," *Microprocessors and Microsystems*, vol. 102, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Sahith Guturu et al., "Design Methodology for Highly Accurate Approximate Multipliers for Error Resilient Applications," *Computers and Electrical Engineering*, vol. 110, pp. 1-8, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Shravani Chandaka, and Balaji Narayanam, "Hardware Efficient Approximate Multiplier Architecture for Image Processing Applications," *Journal of Electronic Testing*, vol. 38, pp. 217-230, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Elham Esmaeili, and Nabiollah Shiri, "An Efficient Approximate Multiplier with Encoded Partial Products and Inexact Counter for Joint Photographic Experts Group Compression," *IET Circuits, Devices & Systems*, vol. 2024, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] Samad Shirzadeh, and Behjat Forouzandeh, "High Accurate Multipliers using New Set of Approximate Compressors," *AEU - International Journal of Electronics and Communications*, vol. 138, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] M. Ramkumar Raja et al., "Energy Efficient Enhanced all Pass Transformation Fostered Variable Digital Filter Design based on Approximate Adder and Approximate Multiplier for Eradicating Sensor Nodes Noise," *Analog Integrated Circuits and Signal Processing*, vol. 118, pp. 399-413, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] S. Harichandra Prasad, and K. Kumar, "Performance Analysis of Low Power Inexact Recursive Multipliers for Image Processing Applications," *Circuits, Systems, and Signal Processing*, vol. 45, pp. 621-646, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Mahdi Taheri et al., "AdAM: Adaptive Approximate Multiplier for Fault Tolerance in DNN Accelerators," *IEEE Transactions on Device and Materials Reliability*, vol. 25, no. 1, pp. 66-75, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Yogeswari Palanisamy et al., "Power-efficient MAC Unit for Image Processing using Dadda Multiplier and Approximate Adder," *Australian Journal of Electrical and Electronics Engineering*, pp. 1-13, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[12] Ayoub Sadeghi et al., "Energy Efficient Compact Approximate Multiplier for Error-Resilient Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 12, pp. 4989-4993, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Bo Liu et al., "Timing Error Tolerant CNN Accelerator With Layerwise Approximate Multiplication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 12, pp. 4412-4425, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Lalit Bandil, and Bal Chand Nagar, "SIEAA: Significant Input Extraction-based Error Optimized Approximate Adder for Error Resilient Application," *Integration*, vol. 101, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[15] Vidya Sagar Potharaju, and V. Saminadan, "Design and Analysis of Low Power Reversible Majority Logic-Based Adder/Subtractor Circuits with Parallel Computing Optimization," *SN Computer Science*, vol. 6, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[16] Maryam Banisharif Dehkordi, and HamidReza Ahmadifar, "A New Approximate (8; 2) Compressor for Image Processing Applications," *IETE Journal of Research*, vol. 70, no. 2, pp. 1352-1360, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] Edris Zaman Farsa, Arash Ahmadi, and Oliver Keszocze, "Reconfigurable Digital FPGA Implementations for Neuromorphic Computing: A Survey on Recent Advances and Future Directions," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 9, no. 5, pp. 3210-3232, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[18] Mohsen Nourazar et al., "Code Acceleration Using Memristor-Based Approximate Matrix Multiplier: Application to Convolutional Neural Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 12, pp. 2684-2695, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[19] Shaghayegh Vahdat et al., "TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1161-1173, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[20] Ladan Sayadi, Somayeh Timarchi, and Akbar Sheikh-Akbari, "Two Efficient Approximate Unsigned Multipliers by Developing New Configuration for Approximate 4:2 Compressors," *IEEE Transactions on Circuits and Systems I: Regular, Papers*, vol. 70, no. 4, pp. 1649-1659, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Muhammad Hamis Haider, and Seok-Bum Ko, "Booth Encoding-Based Energy Efficient Multipliers for Deep Learning Systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 6, pp. 2241-2245, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Dimitrios Danopoulos et al., "AdaPT: Fast Emulation of Approximate DNN Accelerators in PyTorch," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 6, pp. 2074-2078, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Annachiara Ruospo et al., "Investigating Data Representation for Efficient and Reliable Convolutional Neural Networks," *Microprocessors and Microsystems*, vol. 86, pp. 1-42, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[24] U. Anil Kumar et al., "CAAM: Compressor-Based Adaptive Approximate Multiplier for Neural Network Applications," *IEEE Embedded Systems Letters*, vol. 15, no. 3, pp. 117-120, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Zhen Li et al., "Adaptable Approximate Multiplier Design Based on Input Distribution and Polarity," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 12, pp. 1813-1826, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[26] Botang Shao, and Peng Li, "Array-Based Approximate Arithmetic Computing: A General Model and Applications to Multiplier and Squarer Design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 4, pp. 1081-1090, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[27] Faraz Baraati et al., "Efficient and High Accuracy Approximate Multiplier Design Approach for AI Application," *Circuits, Systems, and Signal Processing*, vol. 44, pp. 8635-8656, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[28] Suganthi Venkatachalam et al., "Design and Analysis of Area and Power Efficient Approximate Booth Multipliers," *IEEE Transactions on Computers*, vol. 68, no. 11, pp. 1697-1703, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[29] Mohammad Saeed Ansari et al., "Low-Power Approximate Multipliers Using Encoded Partial Products and Approximate Compressors," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 404-416, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[30] L. Hemanth Krishna et al., "Energy-Efficient Approximate Multiplier Design with Lesser Error Rate Using the Probability-Based Approximate 4:2 Compressor," *IEEE Embedded Systems Letters*, vol. 16, no. 2, pp. 134-137, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Seyed Amir Hossein Ejtahed, and Somayeh Timarchi, "Efficient Approximate Multiplier Based on a New 1-Gate Approximate Compressor," *Circuits, Systems, and Signal Processing*, vol. 41, pp. 2699-2718, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[32] Ahmad Towhidy, Reza Omidi, and Karim Mohammadi, "On the Design of Iterative Approximate Floating-Point Multipliers," *IEEE Transactions on Computers*, vol. 72, no. 6, pp. 1623-1635, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[33] Garima Thakur, Harsh Sohal, and Shruti Jain, "Power–Area-Optimized Approximate Multiplier Design for Image Fusion," *Circuits, Systems, and Signal Processing*, vol. 43, pp. 2288-2319, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[34] Yajuan He et al., "A Probabilistic Prediction-Based Fixed-Width Booth Multiplier for Approximate Computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4794-4803, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[35] Sukanya Balasubramani, Uma Jagadeeshan, and Umapathi Krishnamoorthy, "Performance Optimized Approximate Multiplier Architecture ST-AxM - based on Statistical Analysis and Static Compensation," *Microelectronics Reliability*, vol. 151, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[36] Ying Wu et al., "LMM: A Fixed-Point Linear Mapping Based Approximate Multiplier for IoT," *Journal of Computer Science and Technology*, vol. 38, pp. 298-308, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[37] P. Shareefa Fairoose, and Ashutosh Mishra, "An Efficient Architecture of Truncated Booth Multiplier for AI Application," *Integration*, vol. 106, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[38] Celia Dharmaraj, Vinita Vasudevan, and Nitin Chandrachoodan, "Analysis of Power–accuracy Trade-off in Digital Signal Processing Applications using Low-power Approximate Adders," *IET Computers & Digital Techniques*, vol. 15, no. 2, pp. 97-111, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[39] Bahram Rashidi, "Efficient and Low-cost Approximate Multipliers for Image Processing Applications," *Integration*, vol. 94, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[40] Shyama Gandhi et al., "Approximate Leading One Detector Design for a Hardware-Efficient Mitchell Multiplier," *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, Edmonton, AB, Canada, pp. 1-4, 2019. [CrossRef] [Google Scholar] [Publisher Link]