Proceedings of the 3rd International Conference on Functional Materials and Civil Engineering
DOI: 10.54254/2755-2721/2025.GL30068

Error-Resilient Hardware Design with Approximate Computing: A Review of Circuit-Level and System-Level Optimization Strategies

Yuting Kang[1*†], Weiqiao Chen[2†]

[1]*School of Mechanics and Automation, Shanghai University, Shanghai, China*
[2]*School of Electronic Science and Technology, Xiamen University, Xiamen, China*
**Corresponding Author. Email: kyt0408@shu.edu.cn*
[†]*These authors contributed equally to this work and should be considered as co-first anthors.*

Abstract. Approximate computing (AxC) is a method that allows tiny and controlled errors in exchange for lower power consumption and reduced hardware cost. It offers an effective way to balance computational accuracy and efficiency in error-tolerant tasks. This paper explains two representative designs that use approximation at both the circuit level and the system level. The first design is the Error-Reducing Progressive Prediction Approximate Adder (ERCPAA). It improves accuracy by using progressive carry prediction and constant truncation while maintaining low hardware overhead. The second design is an adaptive Least Mean Square (LMS) filter that uses a sign-aware approximate multiplier. This multiplier lowers computational cost and supports stable convergence. A detailed evaluation of accuracy, power, and area shows how these designs work together to produce efficient and error-resilient hardware.

Keywords: Approximate Computing, Error Resilience, ER-CPAA, LMS Filter, Hardware Efficiency, Circuit Design

1. Introduction

Modern digital systems face the growing demand to maximize performance without sacrificing computational throughput [1,2]. Traditional precise calculation methods often cannot meet these conflicting objectives, prompting the emergence of approximate computation (AxC). By deliberately allowing small, controlled errors, approximate computation generates significant benefits in both energy efficiency improvements and design simplification [3].

Although a wide range of AxC techniques have been explored, much of the existing research remains confined to a single abstraction layer—either low level circuit approximations or high-level algorithmic ones. As a result, the coordination between these layers is often weak. Circuit-oriented designs may deliver efficient yet rigid architectures, whereas system-level approaches risk uncontrolled error propagation. Consequently, a persistent gap exists between local error control and system-aware error resilience [4].

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.GL30068

To address this limitation, this paper explores two representative AxC design approaches that collectively advance integration at both the circuit and system levels. The first type is a circuit-level approximate adder. It integrates two approximation mechanisms, enhancing both energy efficiency and fault tolerance. More specifically, it employs a carry prediction scheme that enhances accuracy by utilizing additional input information.

This design uses a constant truncation method. It cuts hardware cost by handling the least significant bits in a simpler way. The accuracy loss is small. With these features, the Error-Reduced Carry-Predicting Approximate Adder (ERCPAA) reaches a good balance between power use, chip area, and accuracy. It is well-suited for low-power and fault-tolerant systems.

At the system level, the second design adds a sign-aware approximate multiplier to an adaptive LMS filter. The multiplier adjusts its work based on the sign, the size of the input, and the instant error. Even if it use in an approximate math situation, this adaptive behavior will keep the filter stable. It also lowers computation cost, saves power, and reduces chip area without hurting the filter's real performance.

Together, these two designs show different but complementary ways to balance accuracy and efficiency. They also show how strong algorithms and approximate hardware can work together to build energy-efficient and fault-tolerant signal-processing systems. Still, it is not easy to scale these mixed methods to larger tasks and more varied applications. This open problem continues to push for further research.

2. Background

AxC has gained much attention as an effective means to meet the growing need for energy-efficient and high-performance designs, particularly in applications where a certain level of error can be tolerated. This section provides an overview of the fundamental concepts behind AxC and discusses its practical relevance within modern edge-oriented hardware systems.

2.1. Basic principles of approximate computing

Approximate computing allows a system to decrease accuracy in exchange for energy, delay, and silicon area. Many applications can tolerate small amounts of error. Examples include multimedia processing, machine learning, and data mining. Several principles guide approximate computing [3]:

1) Error Resilience: Some applications do not require exact results and tolerate small deviations.

2) Graceful Degradation: Approximation methods avoid sudden or extreme quality loss.

3) Application-Specific Approximation: Designers choose where to approximate based on the error sensitivity of each task.

4) Cross-Layer Approximation: Approximation can be applied at the circuit, architecture, or software level.

5) Design Space Exploration (DSE): Designers explore accuracy, energy, latency, and area to find the best configuration.

6) Quality Metrics: Metrics such as Mean Error Distance (MED), Normalized Mean Error Distance (NMED), and the Measure of Fairness (FoM) measure approximate performance.

7) Hardware Simplification: Approximation removes redundant logic or reduces bit-width.

8) Trade-off Awareness: Designers make choices based on the relationship between accuracy and efficiency.

DOI: 10.54254/2755-2721/2025.GL30068

2.2. Current approximate computing methodology

Approximate computing helps reduce energy use and execution time in data-intensive tasks [5]. It does not replace exact computing entirely. Instead, it introduces controlled errors only in places where these errors are safe. The method starts with analyzing the application to identify operations that tolerate approximation. Designers then apply approximation across software, hardware, or architecture. They use error models to estimate the impact. After that, they explore design space to balance quality with efficiency. They finally test system-level output quality. Methods such as SETA, LOA, and ETAI show how simple logic can reduce area and delay.

In tasks that can accept small errors without losing functional correctness or system reliability, AxC uses a fixed and clear design workflow. The designer first analyzes the application and the workload. This step identifies which computations are critical and which are not. The designer then applies approximation at different abstraction levels. At the software level, this may include precision scaling or skipping selected tasks. At the hardware level, this may include approximate arithmetic units or voltage over-scaling. At the architecture level, this may include approximate accelerators or processors. After that, the designer builds analytical error models. The designer also performs design space exploration to measure the balance between efficiency and output quality. In the final step, the system validates quality-of-service by using domain-specific error metrics. This cross-layer and application-driven method increases performance. It also reduces power use and saves silicon area. At the same time, it keeps the output quality at a level that the target application can accept.

Lee et al. [6] introduced the Simplified Error Tolerant Adder (SETA). It is an improved design that increases energy and power efficiency. It achieves this improvement by modifying the lower-part approximation logic, as shown in Fig. 1. SETA reduces hardware complexity and delay. Its approximate section needs only two logic gate delays. Experiments show that SETA reduces area by 67% and saves power by 91% compared with a conventional Ripple Carry Adder (RCA). SETA also provides better energy efficiency than other approximate adders such as LOA [7] and ETAI [8]. In practical image-processing tasks, the visual quality of the output remains almost unchanged when SETA is used.

However, SETA has a similar error behavior to ETAI and LOA. Errors occur when both input bits in the inaccurate part are equal to '1'. SETA's Normalized Mean Error Distance (NMED) is slightly worse than that of ETAI. This difference shows the trade-off between hardware simplicity and computational accuracy. The maximum error magnitude grows almost exponentially with the number of inaccurate bits, reaching up to $2^n - 1$ for n inaccurate bits. The OR-based simplification also skews the output distribution toward larger values. As a result, both the NMED and the average error metrics degrade when the number of inaccurate bits increases. This behavior highlights the compromise between improved energy efficiency and reduced computational accuracy.

DOI: 10.54254/2755-2721/2025.GL30068

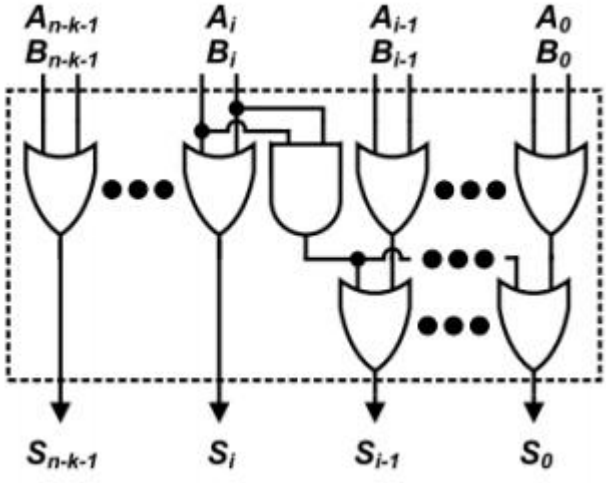


Figure 1. General architecture of SETA [6]

Seo et al. [9] introduced the Hybrid Error Reduction Approximate Adder (HERLOA). The goal of this design is to balance accuracy and energy use. Its main structure is shown in Figure 2. HERLOA cuts the error rate by a large amount compared with earlier adders like LOA [7] and ETAI [8]. It brings the error rate down to about 50%, while the older designs often reach 68% or more. It also improves several accuracy metrics. At the same time, it keeps both area and power overhead under 10%.

HERLOA shows strong efficiency in power, energy, and delay. It can outperform many other approximate adders by up to two times in these areas. It does show slightly worse mean error distance than ECPETA, since ECPETA uses a more advanced carry prediction method. But HERLOA has a simpler structure and uses less power. This makes it useful for error-resilient tasks such as DSP and deep learning. Because it offers both accuracy and efficiency, HERLOA is a strong option for building energy-efficient computing systems.

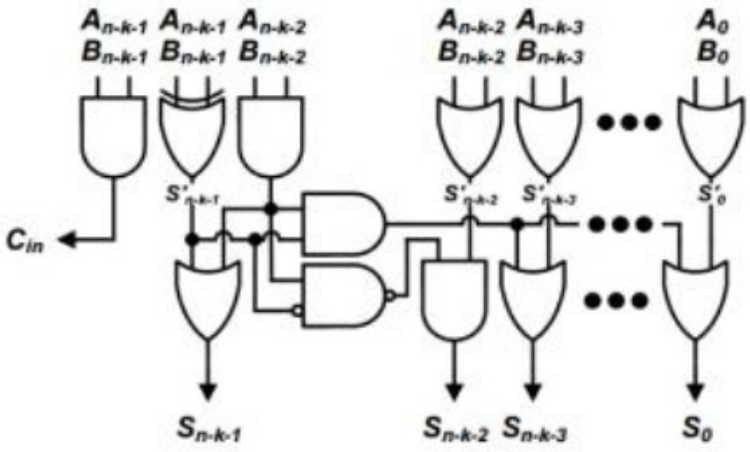


Figure 2. General architecture of HERLOA [7,9]

2.3. Application field

Approximate Computing (AxC) has many promising uses in edge computing hardware, as described by Damsgaard et al. [10]. Figure 3 shows a framework that explains how AxC can be applied in a

DOI: 10.54254/2755-2721/2025.GL30068

systematic way. The process begins by identifying the level of error resilience. This step sets the amount of error the system can tolerate. The next step is to characterize approximate kernels. This step helps select the kernels that are suitable for approximation. After that, the system evaluates these kernels using representative workloads. This evaluation checks whether the kernels work properly for applications such as machine learning, image and video processing, and system reliability tasks. This structured workflow ensures that AxC can be implemented efficiently in edge computing environments.

The system performs Quality Evaluation to achieve output accuracy. It conducts Approximation Tuning to adjust parameters in real-time. It also carries out Dynamic Quality Management to maintain a balance between effectiveness and precision. The diagram shows the continuous AxC process. This process applies approximations to individual components while keeping the overall system stable.

AxC technology is effective in many applications. These include ML models, image processing, and hardware optimization. Integrating AxC with edge architecture enables energy-efficient processing near the data source. It also allows low-latency processing. This approach benefits various sectors. It does not produce high error rates.

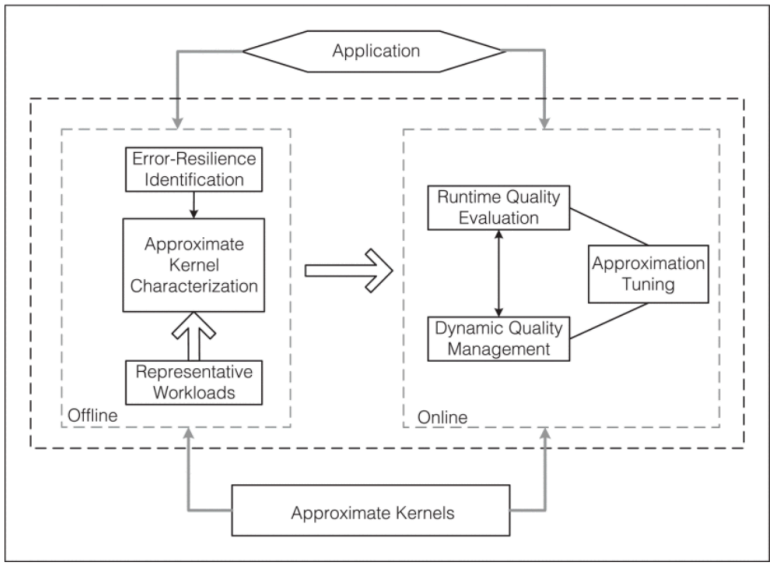


Figure 3. Overall framework of approximate computing

2.4. 4:2 compressors in multiplication

Using compressors in digital multiplication is essential for accumulating partial products. In reference [8], the multiplier architecture highlights the importance of compressors. The 4:2 compressor is particularly important. It plays a key role in improving the performance of arithmetic circuits. This use of compressors is vital for optimizing computational efficiency.

The 4:2-compressor architecture is shown in Fig.4. It combines four input bits. These bits usually represent two separate 2-bit binary values. It also considers a carry-in signal. This produces three output signals. The Sum (S) represents the least significant bit of the sum of all input bits and the carry-in. The Carry (C) acts as an intermediate carry for weighted summation. The Carry-out (Cout) moves to the next compression stages to maintain arithmetic continuity.

The new mechanism shifts four input bits into two output signals: sum and carry. This reduces the number of partial product stages. It also keeps accuracy. The 4:2 compressor outperforms normal

DOI: 10.54254/2755-2721/2025.GL30068

adders. It enables multi-bit reduction. It also reduces carry propagation delays. These features help multiplication structures like Wallace and Dadda trees. Overall, this compression mechanism improves performance in multiplication operations.

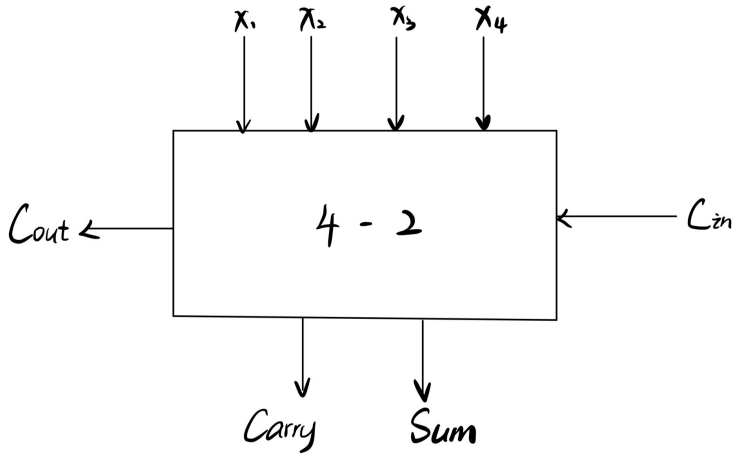


Figure 4. Implementation of 4:2 compressor

3. Error-reduced carry prediction and constant truncation-based approximate adder architecture

ERCPAA decreases error rates while keeping hardware efficient. It balances accuracy and hardware resources. Previous methods often could not achieve this balance. ERCPAA improves efficiency and accuracy in computational tasks. The Efficient Rounding with Carry Prediction and Approximation Adjustment (ERCPAA) is a new method. It combines progressive carry prediction with constant truncation and error reduction. This enhances accuracy and reduces hardware overhead. ERCPAA uses a rounding pre-calculation approach. It lowers carry prediction errors. It also improves overall accuracy by reducing error distance metrics. ERCPAA outperforms existing approximate adders in efficiency. It also has better power and energy consumption. These features make ERCPAA suitable for error-tolerant applications. Examples include digital image processing, like Gaussian smoothing, and machine learning, like k-means clustering. These applications need high accuracy with low hardware cost.

3.1. Overall structure

Fig. 5 shows the approximate adder design. It splits an n-bit input into two parts. One part is a precise k-bit segment. The other part is an imprecise (n — k)-bit portion. The precise segment uses a conventional k-bit adder. This can be a ripple carry adder (RCA) or a carry lookahead adder (CLA). It generates a precise output. It uses the most significant bit (MSB) input and the carry from the imprecise segment. This method makes computation more efficient. It also maintains system accuracy.

Approximate adders use the least significant bit (LSB) input to produce an approximate output and also provide a rounding input for the exact adder. The process has three main parts. The first part is the approximate full adder (FA) cell array, which simplifies the addition and generates both approximate results and rounding values for the higher bits in the inexact section. The second part is the rounding prediction logic, which improves the rounding input for the exact adder by checking

DOI: 10.54254/2755-2721/2025.GL30068

the two most important bits of the inexact part; it requires only two extra logic gates and offers better precision than older methods. The third part is the constant truncation logic with error reduction, which lowers the number of LSBs processed by the exact adder, reduces hardware cost, and sets the LSB outputs to a fixed “0” or “1” to optimize performance. By combining these three parts, approximate adders achieve higher efficiency and accuracy without adding much circuit complexity or hardware cost.

Approximate adders combine these components carefully. They improve efficiency. They also improve accuracy. They do not greatly increase circuit complexity. They do not greatly increase hardware costs.

3.2. Approximate full adder

The proposed ERCPAA [11] adder uses approximate Full Adder (FA) cells, improved carry prediction logic, and constant truncation. It balances accuracy and hardware efficiency. The adder simplifies traditional logic. It uses OR gates instead of XOR gates in approximate FAs for summation. It also eliminates carry chain propagation. This reduces area and critical path delay. The integration improves performance. It also maintains computational precision.

Many existing approximate adders use AND-based rounding prediction. They generate rounding inputs for accurate parts. They use the MSBs of inexact parts, often Ank1 and Bnk1. This method is hardware efficient. However, it has a high prediction error rate of about 25%. Alternative designs like LOA [7] and ETAI [8] do not use rounding prediction. They have error rates of 50% because approximation is poorly controlled. This shows that more precise and reliable rounding prediction methods are needed. They can reduce errors and improve overall performance.

The ERCPAA [11] improves prediction accuracy. It introduces a carry prediction scheme. The scheme does not use only the most significant bit of the inexact part. It uses the (n — k — 1)th bit and the (n — k — 2)th bit. This allows better predictions of carry progression. The approach uses a logic function. It considers multiple bits for more complete prediction analysis.

$$C_{in} = C_{n-k-1} + P_{n-k-1} \bullet C_{n-k-2} \tag{1}$$

The propagation bit Pi = AiBi comes from XOR-based full adders in the imprecise part. It can cause errors. These errors happen when the carry comes from bits at positions (n — k — 3) or lower. The carry then propagates through the (n — k — 2) and (n — k — 1) bits. This reduces prediction accuracy. It occurs because the signal propagation is inaccurate.

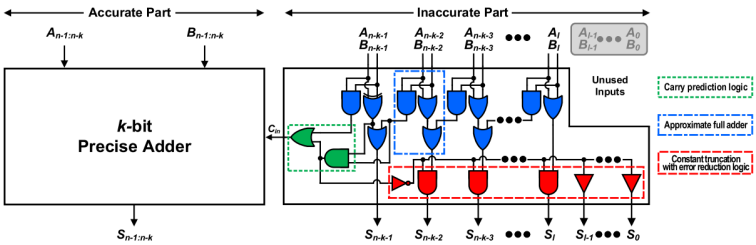


Figure 5. Overall hardware architecture of the cited approximate adder, termed ERCPAA [11]

DOI: 10.54254/2755-2721/2025.GL30068

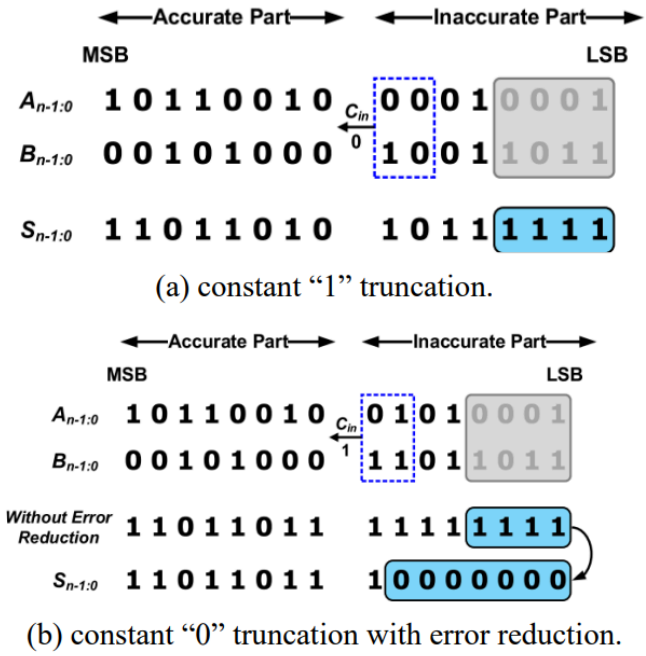


Figure 6. Operations of the proposed adder [11]

Assuming bitwise independence of the input operands, the theoretical error rate for carry prediction, denoted as ERCP(n, k), is given by:

$$ERCP\,(n,k) = P\,(P_{n-k-1} \bullet P_{n-k-2} \bullet E_{ca}) = \tfrac{1}{2} \bullet \tfrac{1}{2} \bullet \left(1 - \tfrac{1}{2^{n-k-2}}\right) \quad (2)$$

where Eca represents the event when a carry is generated in the lower bits. Experiments validate this model. They show a prediction error rate of 12.305%. This is a significant improvement over traditional AND-based or non-predictive rounding methods.

ERCPAA [11] uses constant truncation and error reduction logic. It balances accuracy and efficiency. It fixes the output of the LSBs to a constant value, such as "1." This avoids unnecessary logic computation. It also reduces processing time. The system operates effectively. Error reduction logic activates only when specific signals align. This prevents inaccuracies from approximate summation exceeding the accurate value. Outputs in the imprecise range are set to "0" instead of "1," except for the most significant bit. This keeps the approximation close to the true value. It improves overall system accuracy.

The effectiveness of the proposed method was validated using a 16-bit adder configured with parameters n = 16, k = 8, and l = 4. In the case where traditional constant truncation is applied, fixing the least significant l bits to "1" resulted in an Error Detection (ED) value of 211 for a given input. However, when utilizing ERCPAA [11] error reduction logic, the ED value dropped significantly to 84. This demonstrates a substantial reduction in error achieved through the adaptive correction strategy, with a maximum potential ED reduction of 2n-k-1 — 1. These results highlight the practical benefits provided by the proposed approach.

3.3. Performance

ERCPAA [11] is an important design. It offers good computational accuracy and high hardware efficiency. It uses carry prediction and truncation to advance performance. These techniques help

DOI: 10.54254/2755-2721/2025.GL30068

increase accuracy and reduce hardware costs. In a 16-bit design with equally precise and approximate parts where k equals 8, the carry prediction unit shows a low error rate of 12.3%. In tests, the ED value decreases from 211 to 84. With tuned parameters, ERCPAA [11] reaches a NMED of 0.974×10^{-3} and a MRED of 1.22×10^{-2}. These results show that ERCPAA [11] is a strong candidate for future approximate adder designs.

ERCPAA [11] uses hardware resources very efficiently. Its area is only 128.2 μm², which is much smaller than RAP-CLA (333.8 μm²) and close to HERLOA [7,9] at 131.7 μm². Its power cost is 35.6 μW, which is 1.90× lower than a standard RCA (58.5 μW). ERCPAA [11] also reduces power-NMED by 95.7% and cuts energy-NMED by 91.1% compared with RAP-CLA. These results show that it offers a strong balance between accuracy and resource savings. ERCPAA [11] works well in error-tolerant tasks, such as image processing with Gaussian smoothing and machine learning with k-means clustering. It keeps good output quality even with approximation.

3.4. Discussion of the proposed methods

ERCPAA [11] exhibits better performance than earlier approximate adders, such as SETA [6] and HERLOA [7,9]. It offers higher accuracy and better energy efficiency. Its key feature is an improved carry prediction unit. This unit uses the two most significant bits in the approximate segment. It lowers the error rate to 12.3%, which is a clear improvement over the 25% error rate of simple AND-based prediction. This refinement increases precision and reduces energy cost.

ERCPAA [11] works well in many error-tolerant tasks. Its methods expand the accurate computation range without adding much power or area overhead. Tests show strong gains: NMED improves by up to 91.4%, and MRED improves by 99.4%. These results show that ERCPAA [11] offers a better accuracy–efficiency tradeoff than SETA [6] and HERLOA [7,9]. It also uses conditional correction logic for low-bit patterns, which helps reduce error distance more effectively than fixed-LSB truncation. It keeps high computational precision while keeping power and area low.

ERCPAA [11] has clear benefits, but its design is more complex than lightweight adders such as SETA [6]. HERLOA [7,9] also uses hybrid error-reduction logic and can reach lower latency in some cases because it has less hardware overhead. ERCPAA can match these designs in energy efficiency and error performance, but it does not reduce area as much as SETA or HERLOA because of the extra logic blocks. SETA has very low delays because of its small two-gate inaccurate segment. ERCPAA relies on XOR, AND, and OR logic, which increases the critical path and raises the overall delay. ERCPAA's accuracy evaluation is based on uniformly random inputs. This assumption may not hold for real workloads, such as structured image data or formatted network traffic.

4. A high accuracy and hardware efficient filter design with approximate computing

4.1. Innovative approximate computing scheme

Xie et al. [12] proposed an approximate computing method that is specifically designed for Least Mean Squares (LMS) adaptive filters. They wanted to reduce hardware cost and complexity compared with conventional designs. The LMS filter is one of the most commonly used adaptive algorithms. Because it is simple to implement and shows stable convergence. Additionally, It plays an important role in noise cancellation, system identification, and echo suppression.

In Fig. 8, the LMS filter comprises two important components. Firstly, a FIR block that could control the input signal. Second, a weight-update module that adjusts the filter coefficients in

DOI: 10.54254/2755-2721/2025.GL30068

response to the error signal. This feedback loop makes the filter be able to refine its output error, and makes it well-suited for the deployment of approximate computing techniques.

Exploiting on this inherent property, Xie et al. developed a system-level approximation strategy centered on directional error control rather than the minimization of overall average error. Their approximate multiplier is designed to deliberately produce negative errors for positive inputs and positive errors for negative inputs. This fixed error direction facilitates inter-iteration error cancellation while curbing the accumulation of noise throughout operational cycles.

The success of this cancellation depends on the input's statistical nature. Theoretical analysis confirms the mechanism works for all input sign combinations. It performs best when the signal has zero mean. Under these conditions, positive and negative errors balance out over time. Experiments with white Gaussian input validated this. When the input maintains the same sign, the cancellation effect weakens. Even so, the proposed multiplier's predictable behavior boosts convergence stability. This outperforms random-error designs.

All in all, this method demonstrates a practical way to apply approximate computing to adaptive filters while maintaining reliable and efficient system performance.

4.2. Optimized low-complexity 4:2 compressor

Xie et al. [12] proposed a novel approximate 4:2 compressor to materialize their approximation scheme. A 4:2 compressor is a fundamental logic unit that compresses four input bits into two output bits, widely used to accelerate partial product accumulation in multipliers. Their design significantly deviates from the conventional accurate compressor by eliminating the Carry-in (Cin) and Carry-out (Cout) signals. Furthermore, the authors deliberately modified the logic function (e.g., by simplifying the truth table) to ensure that the introduced approximation errors are predominantly negative. This deliberate error bias is a key feature of their design, as it is strategically intended to facilitate system-level error compensation by counteracting positive errors generated elsewhere in the approximate multiplier, particularly for negative inputs.

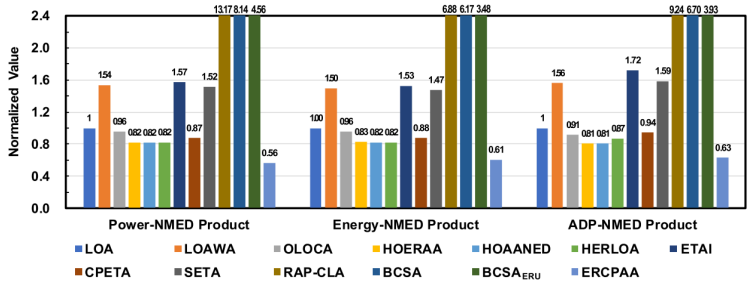


Figure 7. Normalized Power-Normalized Mean Error Distance (NMED), energy-NMED, and Area-Delay Product (ADP)-NMED products of approximate adders [11]

DOI: 10.54254/2755-2721/2025.GL30068

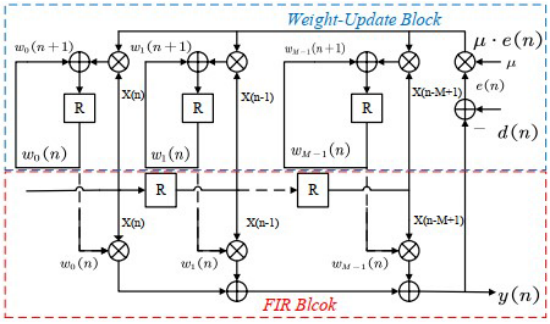


Figure 8. Adaptive LMS filter-hardware overview [12]

As was shown in Fig.9, the proposed gate-level imple-mentation significantly reduces the number of logic gates, leading to notable hardware savings. Although this increases the approximation error, the system compensates for it through sign-bit correction, which adjusts the most significant bit of the result to ensure that the sign of the output remains consistent with the expected value. This mechanism helps maintain overall computational accuracy despite the reduced precision.

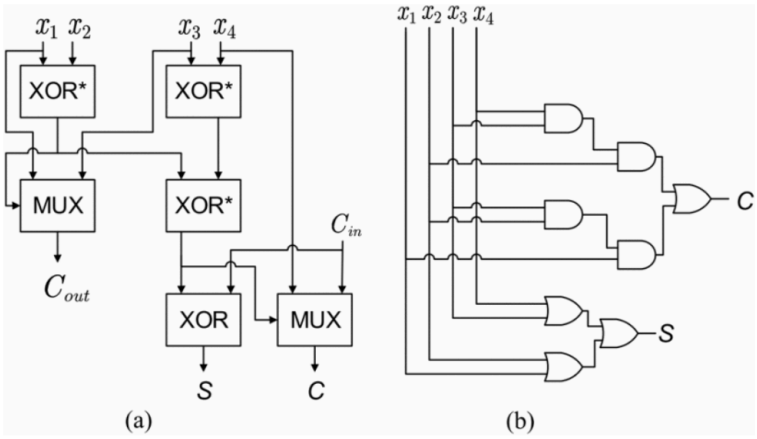


Figure 9. 4:2 compressor designs: (a) accurate 4:2 compressor (XOR* is an XOR-XNOR module), (b) approximate 4:2 compressor gate circuit [12]

4.3. Precision-optimized approximate multiplier architecture

This section introduces a signed approximate multiplier. It is built on the proposed approximate 4–2 compressor and aligned with the overall approximation strategy. Figure 10 shows that the 12-bit multiplier uses a two-stage partial product compression structure. To improve computational efficiency, an advanced encoding scheme is adopted. It cuts down the number of partial product rows, streamlines the compression tree and trims hardware complexity.

The accumulation stage is divided into three main regions and a compensation mechanism:

Accurate Region: A 12-bit exact segment from S1 to S1 [12] and 11-bit from S2 [13] downward;

Approximate Region: A 2-bit section from S1 [10] to S1 [11] and S2 [11] to S2 [12], where approximate compressors are applied;

DOI: 10.54254/2755-2721/2025.GL30068

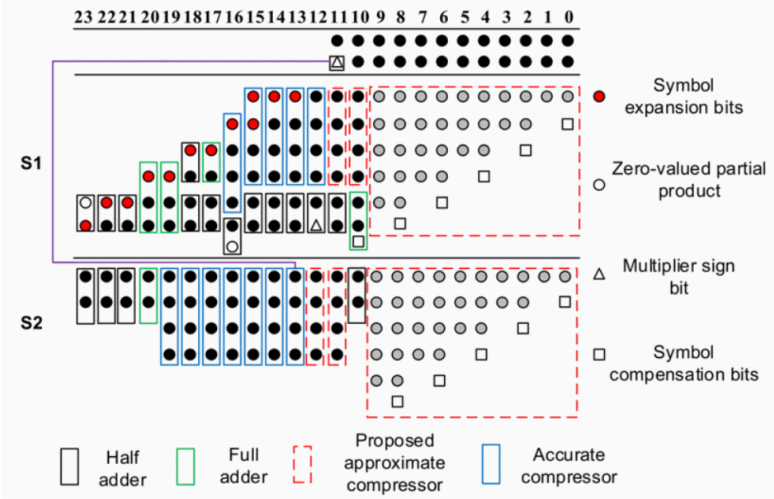


Figure 10. Partial product of 12 x 12 bit Approximate Multiplier used in FIR Block (AMF) [12]

Truncated Region: A 10-bit portion that is discarded to save hardware.

Compensation Mechanism: The sign bit of the multiplier is used to replace the carry-in (Cin) of an exact 4:2 compressor in the lowest valid column (S2 [13]) of the accurate region;

Partial Product Adjustment: The partial product at S1 [12] is replaced by the multiplier's sign bit.

This compensation approach not only reused the sign bit to maintain accuracy but also didn't add extra logic gates, thereby keeping hardware overhead minimal.

4.4. Implementation results

This section verifies the proposed approximate multiplier adheres to the intended design strategy. Theoretical analysis—rooted in partial product probabilities and expected error patterns—confirms the multiplier produces directionally biased errors. These errors hinge on the sign bits of the inputs. This trait carries significance: it enables effective error correction at the system level.

The overall error is small and remains under control. This is achieved through a mix of approximate compression, truncation, and compensation steps. The compression stage adds small negative errors, while the compensation step provides a positive offset. Together, they keep the total error low and predictable.

1) Application: The performance of the approximate LMS adaptive filter is tested in a system identification task. As shown in Fig. 11, the filter learns to model an unknown system by reducing the difference between its output y(n) and the target signal d(n). The reference system is an IIR Butterworth low-pass filter with a 1 dB passband gain (0–0.2π rad/sample) and a stopband attenuation of –60 dB at 0.3125π rad/sample. The adaptive filter uses 60 taps, showing that the proposed method can achieve high-quality signal processing with efficient hardware.

2) Accuracy Performance: The accuracy of the proposed design is evaluated using the Mean Squared Error (MSE) metric. As summarized in Table I, the proposed multiplier achieves an MSE of 1.606 × 10-3, which is competitive with the error-recovery designs by Stroll et al [13]. (1.489 × 10-3) and Ha et al [14]. (4.118 × 10-3), and substantially superior to those of Zhu et al [15]. (26.24 × 10-3) and Qian et al [16]. (66.67 × 10-3). The inferior performance of the latter two is mainly due to their use of approximate Booth encoders and less refined compressors, which introduce larger and less predictable errors.

DOI: 10.54254/2755-2721/2025.GL30068

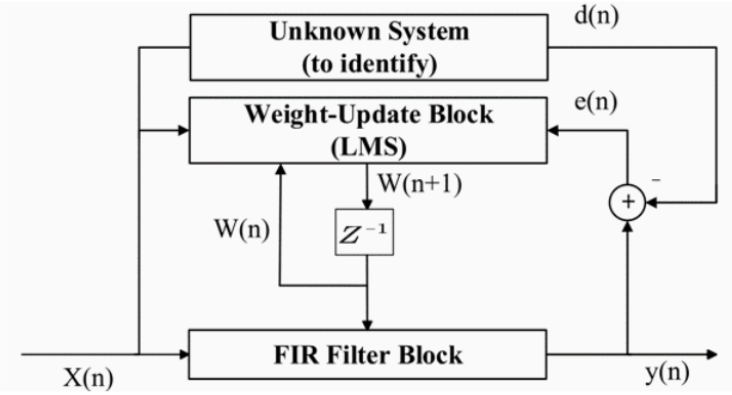


Figure 11. System identification application [12]

3) Hardware Efficiency: All designs were synthesized in a 28 nm CMOS process to evaluate hardware efficiency. The proposed filter achieves the highest reductions in area (15.10%) and power (17.29%) relative to the exact baseline implementation. By comparison, Stroll et al. [13] and Ha et al. [14] report area savings of 13.28% and 14.52%, and power savings of 14.13% and 14.28%, respectively. Designs from Zhu et al. [15] and Qian et al. [16] yield more modest improvements, with approximately 12% area and 11% power reduction. The superior hardware efficiency of the proposed approach stems from its low-complexity approximate 4:2 compressor and a novel sign-aware compensation mechanism, which together enable precise error direction control without compromising system-level performance.

Table 1. Performance of LMS adaptive filters employing different 12×12 approximate multipliers [12]

Employed Multipliers in FIR	MSE [10-3]	Area[μm2]	Power[mW]
Exact	—	80909.39	3.2154
Stroll [13]	1.489	70161.08 (-13.28%)	2.7612 (-14.13%)
Ha [14]	4.118	69160.39 (-14.52%)	2.7564 (-14.28%)
Zhu [15]	26.24	70481.12 (-12.18%)	2.8400 (-11.68%)
Qian [16]	66.67	71575.56 (-11.54%)	2.8488 (-11.40%)
Proposed AMF	1.606	68689.91 (-15.10%)	2.6595 (-17.29%)

5. Conclusion

This paper reviewed two representative approximate computing designs that address the trade-off between computational accuracy and hardware efficiency. At the circuit level, the ERCPAA demonstrated superior error-distance reduction and power efficiency through progressive carry prediction and conditional truncation. At the system level, a sign-aware approximate multiplier integrated into an LMS adaptive filter exploited directional error compensation to enhance convergence stability while lowering area and power consumption.

These two designs together show how different methods can work at separate levels to achieve reliable error handling. The results demonstrate that approximate computing helps lower energy use and chip area without a serious loss of performance. This approach is therefore suitable for building

DOI: 10.54254/2755-2721/2025.GL30068

future systems that are both low-power and error-tolerant. Future studies can apply these ideas to more application domains and explore cross-layer design methods that improve system scalability and reliability.

Acknowledgment

We sincerely thank Professor Bill Nace of Carnegie Mellon University for his guidance throughout the preparation of this literature review. His lectures and insights helped us gain a deeper understanding of the topic and greatly improved the quality of our work. We are also grateful to our teaching assistant Sun Han. His helpful comments and suggestions on the writing process really assist us.

Chen Weijiao and Kang Yuting have made equal contributions to this work and should be regarded as the co-first authors.

References

[1] Yanxiang Huang. “Cross-Layer Optimization for Power-Efficient and Robust Digital Circuits and Systems”. In: arXiv preprint arXiv: 1712.03948 (2017).
[2] Edwin Sha et al. “Power efficiencyfor hard-ware/software partitioning with time and area constraints on mpsoc”. In: International Journal of Parallel Programming 43.3 (2015), pp. 381–402.
[3] Vasileios Leon et al. “Approximate computing survey, part i: Terminology and software & hardware approximation techniques”. In: ACM Computing Surveys 57.7 (2025), pp. 1–36.
[4] Jianping Zeng et al. “Soft Error Resilience at Near-Zero Cost”. In: ICS ’24. Kyoto, Japan: Association for Computing Machinery, 2024. ISBN: 9798400706103. DOI: 10.1145/3650200.3656605. URL: https: //doi.org/10.1145/3650200.3656605.
[5] Ayad M Dalloo et al. “Approximate computing: Concepts, architectures, challenges, applications, and future directions”. In: IEEE Access (2024).
[6] Jungwon Lee et al. “Approximate adder design with simplified lower-part approximation”. In: IEICE Elec-tronics Express 17.15 (2020), pp. 20200218–20200218.
[7] H. R. Mahdiani et al. “Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications”. In: IEEE Transactions on Circuits and Systems I: Regular Papers 57.4 (2010), pp. 850–862. DOI: 10.1109/TCSI.2009.2027626.
[8] Ning Zhu et al. “Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing”. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems 18.8 (Aug. 2010), pp. 1225–1229. DOI: 10 . 1109/ TVLSI . 2009 . 2020591.
[9] Hyoju Seo, Yoon Seok Yang, and Yongtae Kim. “De-sign and analysis of an approximate adder with hybrid error reduction”. In: Electronics 9.3 (2020), p. 471.
[10] Hans Jakob Damsgaard, Aleksandr Ometov, and Jari Nurmi. “Approximation opportunities in edge comput-ing hardware: A systematic literature review”. In: ACM Computing Surveys 55.12 (2023), pp. 1–49.
[11] Jungwon Lee et al. “A Novel Approximate Adder Design Using Error Reduced Carry Prediction and Constant Truncation”. In: IEEE Access 9 (2021), pp. 119939–119953. DOI: 10 . 1109 / ACCESS . 2021 . 3108443.
[12] Wenzhuo Xie et al. “A High Accuracy and Hard-ware Efficient Adaptive Filter Design with Approximate Computing”. In: 2023 IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS). Aug. 2023, pp. 982–986. DOI: 10 . 1109/ MWSCAS57524 . 2023.10405850.
[13] Antonio GM Strollo et al. “Low-power approximate multiplier with error recovery using a new approximate 4-2 compressor”. In: 2020 IEEE International Sympo-sium on Circuits and Systems (ISCAS). IEEE. 2020, pp. 1–4.
[14] Minho Ha and Sunggu Lee. “Multipliers with approxi-mate 4–2 compressors and error recovery modules”. In: IEEE Embedded Systems Letters 10.1 (2017), pp. 6–9.
[15] Yuying Zhu et al. “A probabilistic error model and framework for approximate booth multipliers”. In: Pro-ceedings of the 14th IEEE/ACM International Sympo-sium on Nanoscale Architectures. 2018, pp. 7–12.
[16] Liangyu Qian et al. “Design and evaluation of an approximate Wallace-Booth multiplier”. In: 2016 IEEE international symposium on circuits and systems (IS-CAS). IEEE. 2016, pp. 1974–1977.