

Frontier-Aware Compensated Truncated Multiplier for Energy-Efficient Edge-AI VLSI Datapaths

Dr. T. Satyanarayana¹, Dr. Malothu Amru², Dr. Bitla Prabhakar³, Dr. Somala Rama Kishore⁴

Associate Professor, Dept. of Electronics and Communication Engineering^{1,4}

Professor, Dept. of Electronics and Communication Engineering^{2,3}

CMR Engineering College, Hyderabad, TS, India^{1,2,4}

JNTUH University College of Engineering Jagtial, TS, India³

satyant234@gmail.com, bprabhakar2008@gmail.com.

malothuamru@gmail.com, ramki430@gmail.com

Abstract: Approximate multiplication is attractive for VLSI accelerators used in image processing, digital signal processing, and edge artificial intelligence because many workloads can tolerate bounded numerical error in exchange for reduced switching and arithmetic complexity. This work presents a frontier-aware compensated truncated multiplier (FA-CTM) that removes the lowest partial-product columns while preserving only the boundary, or frontier, column for a compact activity estimator. The frontier population is mapped to a quantized correction term that is added to the retained partial-product sum. Unlike uncompensated truncation, the proposed method targets the systematic negative bias produced by discarded partial products without reconstructing the full lower array. Exhaustive evaluation of all 65,536 unsigned 8-bit input pairs shows a mean error distance of 3.055, normalized mean error distance of 4.70×10^{-5} , maximum error distance of 13, and mean relative error distance of 1.92×10^{-3} . Relative to simple four-column truncation, FA-CTM reduces MED and NMED by 75.1%, MRED by 65.9%, and maximum error by 73.5%. In a 64-term multiply-accumulate benchmark, the mean relative output error falls from 0.0763% to 0.0132%. The partial-product generation domain uses 58 terms instead of 64 and exhibits a 9.35% reduction in measured partial-product toggle events under random operands. A 16-bit sampled validation further shows that frontier compensation remains effective when the truncation boundary is scaled. These results indicate that low-cost error-shaping around the truncation frontier is a practical design direction for energy-conscious VLSI datapaths.

Keywords: approximate computing, approximate multiplier, truncated multiplier, VLSI arithmetic, edge AI, low-power datapath, error-tolerant computing.

I. INTRODUCTION

Power and energy efficiency remain central constraints in modern VLSI systems, particularly in always-on edge devices in which arithmetic datapaths execute large numbers of multiply and multiply-accumulate operations. Conventional exact arithmetic treats every output bit as equally important, even when the surrounding application—such as filtering, recognition, multimedia processing, or machine-learning inference—has statistical or perceptual tolerance to small numerical deviations. Approximate computing exploits this tolerance by deliberately relaxing arithmetic exactness to reduce circuit cost [1], [2]. Multiplication is a particularly important target because partial-product generation and reduction contribute substantial switching activity, logic depth, and layout complexity in DSP and AI datapaths.

Early approximate multiplier research demonstrated that accuracy can be traded for power by simplifying small multiplier blocks [3]. Subsequent architectures explored localized carry propagation with configurable error recovery

[4], approximate 4:2 compressors [5], dynamic-range selection [6], approximate Booth recoding [7], operand rounding [8], approximate radix-4 Booth encoding [9], encoded partial products [10], and compressor-selection strategies [11]–[14]. These studies collectively establish that no single approximation style dominates across all accuracy, signedness, technology, and workload requirements. In particular, compressor-heavy approaches can offer excellent hardware efficiency but require carefully selected approximate cells, while operand-domain approaches may introduce input-dependent bias or larger worst-case errors.

Truncation is attractive because its hardware interpretation is direct: lower weighted partial products can be omitted. The difficulty is that the omitted region is not random. Its value is nonnegative, and discarding it produces a systematic negative error. A constant correction can remove average bias but cannot track operand-dependent variation. The key observation motivating this work is that much of the variation in the omitted region is correlated with the activity of the last discarded diagonal, referred to here as the frontier column. Retaining only those frontier partial-product signals for a small population estimator provides a compact indication of the magnitude of the omitted contribution.

Accordingly, this paper proposes a frontier-aware compensated truncated multiplier (FA-CTM). For an 8×8 unsigned multiplier with a four-column truncation boundary, partial products of weight below 2^4 are excluded from normal reduction. The four frontier terms with index sum $i+j=3$ are retained only as estimator inputs. Their population count f selects one of five quantized correction values. The correction is then combined with the retained partial-product sum. The design therefore avoids reconstructing the full low-significance array while shaping both error magnitude and bias. The main contributions are: (1) a simple frontier-based model that links omitted partial-product energy to the activity of the boundary diagonal; (2) a quantized compensation rule that requires only a small population counter and correction decoder; (3) exhaustive error characterization for all unsigned 8-bit input pairs; (4) application-level evaluation using multiply-accumulate and image-scaling workloads; and (5) a scalability check using 500,000 randomly sampled unsigned 16-bit operand pairs. Hardware conclusions in this manuscript are deliberately reported as architectural counts and switching-domain measurements rather than post-layout silicon measurements; technology-dependent synthesis and physical-design results should be added before fabrication-level claims are made.

II. RELATED WORK

Han and Orshansky described approximate computing as an energy-efficient design paradigm in which controlled inaccuracy is introduced at multiple abstraction levels [1]. Xu et al. later organized the field across programming, architecture, and circuit techniques and emphasized the need to quantify both quality loss and hardware benefit [2]. Within arithmetic, Kulkarni et al. used an underdesigned 2×2 block to construct approximate multipliers and demonstrated substantial power savings in error-resilient applications [3]. Liu et al. proposed configurable partial error recovery to limit the carry-propagation cost during partial-product accumulation [4].

Compressor-based multiplication has received sustained attention. Momeni et al. introduced approximate 4:2 compressor structures and showed how different placements affect power, delay, transistor count, and image quality [5]. Ansari et al. used encoded partial products and approximate compressors to improve the power-delay versus error tradeoff [10]. Esposito et al. developed new approximate compressors and a systematic multiplier construction method [11]. Strollo et al. compared and extended a broad set of 4:2 compressor topologies and showed that the preferred cell depends on operand size, signedness, approximation level, and error metric [12]. Sayadi et al. later proposed probability-aware configurations of approximate compressors for unsigned multipliers [14].

Alternative architectures reduce multiplication complexity without relying exclusively on compressors. DRUM selects a dynamic significant range of each operand and was designed to produce an approximately unbiased error distribution [6]. Jiang et al. accelerated radix-8 Booth multiplication by approximating the preliminary generation of odd multiples [7], while Zendegani et al. proposed RoBA, which rounds operands near powers of two to simplify multiplication [8]. Liu et al. designed approximate radix-4 Booth multipliers with tunable approximation factors [9]. Evolutionary and formally constrained libraries, including EvoApprox8b and later scalable construction methods, demonstrate another route to systematic design-space exploration [13], [16].

The present work differs from these approaches in its design objective. Rather than replacing reduction-tree cells or approximating an entire operand, FA-CTM preserves the high-significance multiplication structure and applies approximation only to a triangular low-weight region. The frontier of that region is observed but not fully summed. This creates a small side channel that estimates the dropped contribution and compensates the dominant truncation bias.

Table I. Representative pre-2024 approximate multiplier literature

Ref.	Year	Main approach	Evaluation focus	Relevance to this work
[3]	2011	Inaccurate 2×2 building block	Power–quality tradeoff	Circuit-level approximation
[4]	2014	Limited carry + error recovery	28-nm power and delay	Configurable compensation
[5]	2015	Approximate 4:2 compressors	Power, delay, transistor count	Reduction-tree approximation
[6]	2015	Dynamic significant range	Unbiased error distribution	Bias control
[7]	2016	Approximate radix-8 Booth recoding	Delay/power and FIR	Recoding approximation
[8]	2017	Operand rounding (RoBA)	DSP speed and energy	Operand-domain approximation
[10]	2018	Encoded partial products + compressors	PDP–error tradeoff	Partial-product design
[12]	2020	Compressor comparison/extension	28-nm 8/16-bit study	Topology selection
[14]	2023	Probability-aware compressors 4:2	28-nm multiplier study	Input-probability awareness

III. PROPOSED FRONTIER-AWARE COMPENSATED TRUNCATED MULTIPLIER

A. Exact and Truncated Partial-Product Model

For unsigned n -bit operands A and B , binary multiplication can be expressed as a weighted sum of partial products $p_{ij} = a_i b_j$. The exact product is

$$P = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (a_i \cdot b_j) \cdot 2^{-j} \quad (1)$$

Let k denote the truncation boundary. A conventional column-truncated multiplier discards every partial product whose weight index satisfies $i+j < k$ and retains the remaining terms. Its output is

$$P_T = \sum_{i+j \geq k} (a_i \cdot b_j) \cdot 2^{-j} \quad (2)$$

The discarded contribution D is therefore nonnegative and equals $P - P_T$. Since $E[P_T - P]$ is negative for uniformly distributed operands, straightforward truncation is biased downward. The maximum dropped value grows with k and with the number of active partial products in the discarded triangle.

B. Frontier Observation

The frontier is defined as the highest-weight discarded diagonal, $i+j=k-1$. For $k=4$, this diagonal contains four terms: $a_0 b_3$, $a_1 b_2$, $a_2 b_1$, and $a_3 b_0$. Their population count is

$$f = \sum_{i+j=k-1} (a_i \cdot b_j), \quad 0 \leq f \leq k \quad (3)$$

Exhaustive analysis of the 8-bit input space shows a strong relationship between f and the conditional mean of D . For $k=4$, the conditional mean omitted contributions for $f=0,1,2,3,4$ are approximately 1.89, 11.78, 22.93, 35.33, and 49.0, respectively. This monotonic structure motivates a low-resolution correction table rather than a general multiplier or large lookup memory.

C. Quantized Compensation

The proposed 8-bit implementation uses the quantized correction map $C(f) = \{0, 8, 24, 32, 48\}$ for $f = \{0,1,2,3,4\}$. These values are close to the conditional means but are chosen on an 8-unit grid to simplify correction generation and alignment. The approximate output is

$$\hat{P} = P_T + C(f) \quad (4)$$

The correction can be implemented as a small combinational population counter followed by decoding of a few constant patterns. Because the frontier signals are used only by the estimator, the lower columns do not require a

conventional reduction tree. The estimator path can be evaluated in parallel with high-order partial-product accumulation and merged near the final adder.

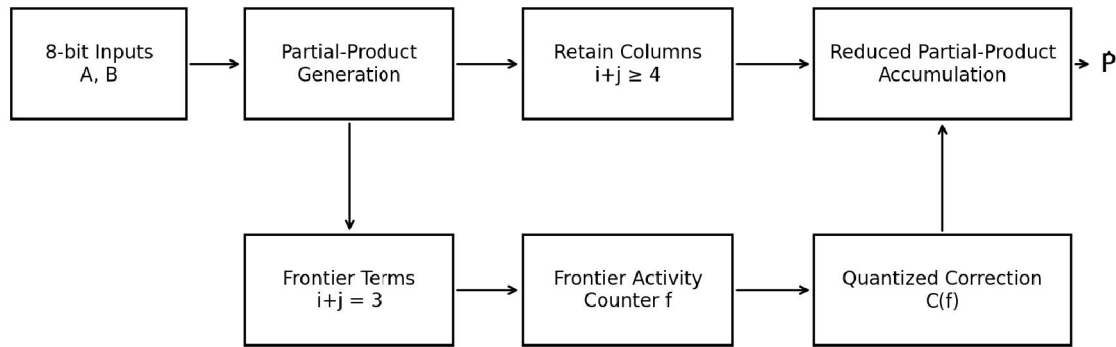


Fig. 1. Proposed FA-CTM architecture: low-weight products are removed from the main reduction network, while the frontier diagonal drives a compact correction path.

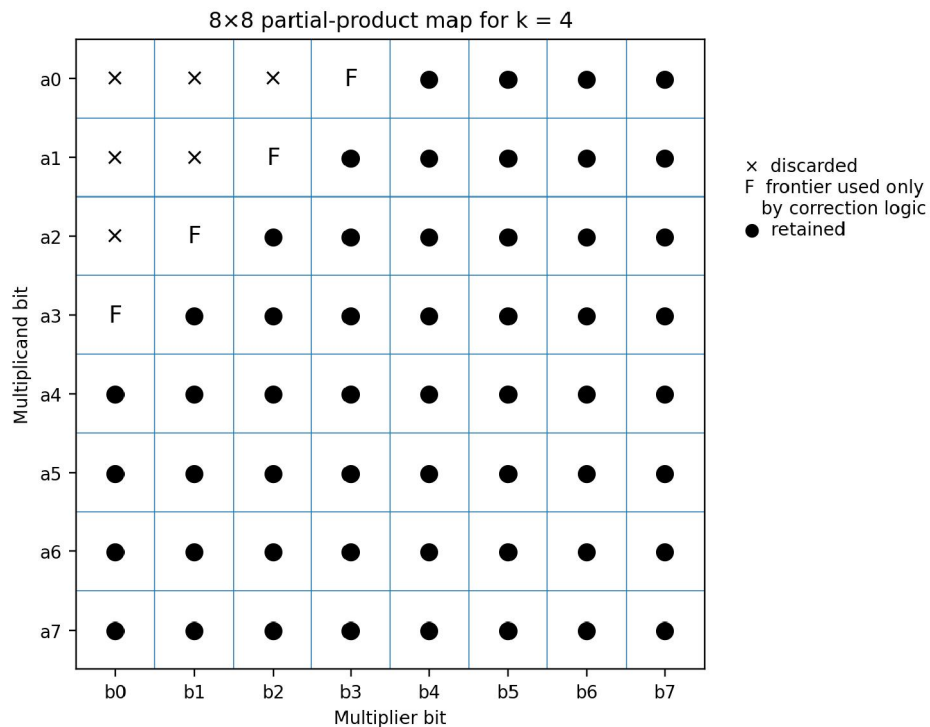


Fig. 2. Partial-product partition for the 8×8, k=4 configuration. The frontier diagonal is observed for compensation rather than fully accumulated as part of the low-order array.

D. Complexity Interpretation

An exact 8×8 unsigned multiplier contains 64 elementary partial-product terms. Four-column truncation removes 10 terms, leaving 54 for the main reduction network. FA-CTM reintroduces only the four frontier terms as estimator inputs, giving 58 generated partial-product signals. Thus, the partial-product generation count is reduced by 9.375% relative to the exact 64-term array. This count does not include the small counter/decoder and should not be interpreted as a complete post-synthesis area percentage; it is an architecture-level measure of the multiplication fabric that is directly removed.

To estimate activity in the same domain, 100,000 random operand pairs were applied as a sequence and transitions were counted on the generated partial-product signals. The exact 64-signal array produced an average of 24.018 signal toggles per input transition, whereas the 58-signal proposed set produced 21.764, a reduction of 9.39%. This metric isolates partial-product generation activity and does not include switching in the final accumulation or correction logic.

IV. EXPERIMENTAL METHODOLOGY

The proposed design was evaluated at arithmetic, application, and architectural levels. All arithmetic results for the 8-bit design were generated by exhaustive enumeration of $A, B \in [0, 255]$, resulting in 65,536 input pairs. Three approximate baselines were considered: uncompensated four-column truncation, a constant-compensated truncation that adds 12 to every truncated product, and FA-CTM. The constant value 12 is close to the unconditional mean omitted contribution (12.25) and therefore provides a useful bias-correction baseline.

For each design, error rate (ER), mean error distance (MED), normalized mean error distance (NMED), mean relative error distance (MRED), maximum error distance (MaxED), and mean signed bias were computed. With P denoting the exact result and $\hat{P}$ the approximate result, the metrics are defined as follows:

$$ED = |P - \hat{P}| \quad (5)$$

$$MED = (1/N) \sum ED \quad (6)$$

$$NMED = MED / P_{\max} \quad (7)$$

$$MRED = (1/N \sum_{P \neq 0}) \sum |P - \hat{P}| / P \quad (8)$$

where $P_{\max} = (2^8 - 1)^2 = 65,025$ for the exhaustive 8-bit experiment. In addition, 10,000 independent 64-term dot products were generated using uniformly random 8-bit operands to evaluate error accumulation in a MAC-like workload. Image-level behavior was examined by applying a fixed-point brightness scaling coefficient of 181/255 to a deterministic 256×256 grayscale test pattern. The approximate products were rounded and rescaled in the same manner as the exact products; image quality was reported using peak signal-to-noise ratio (PSNR) and mean absolute pixel error.

To examine scalability, a 16-bit configuration with $k=8$ was evaluated using 500,000 random operand pairs. The frontier contains eight terms and selects a nine-entry correction table quantized to multiples of 64: {64, 256, 448, 640, 832, 1088, 1280, 1536, 1792}. This experiment is a sampled functional validation rather than a technology-specific synthesis result.

V. RESULTS AND DISCUSSION

A. Exhaustive 8-bit Arithmetic Accuracy

Table II. Exhaustive 8-bit arithmetic results

Design	ER (%)	MED	NMED	MRED	MaxED	Bias	PP cells
Exact	0.00	0.0000	0.000e+00	0.000e+00	0	0.000	64
Truncated-4	81.25	12.2500	1.884e-04	5.641e-03	49	-12.250	54
Constant-compensated	90.62	7.8516	1.207e-04	3.919e-03	37	-0.250	54
Proposed FA-CTM	68.75	3.0547	4.698e-05	1.922e-03	13	-2.125	58

Uncompensated truncation has $MED=12.25$ and a mean signed bias of -12.25 , confirming that all discarded partial products push the output downward. Adding the unconditional constant 12 nearly eliminates average bias, but its MED remains 7.85 and its error rate increases because a correction is injected even when the dropped contribution is small. FA-CTM produces the strongest overall error profile. Compared with simple truncation, MED and NMED fall by 75.1%, MRED falls by 65.9%, and MaxED falls by 73.5%. The proposed error rate also decreases from 81.25% to 68.75%.

The residual FA-CTM bias is -2.125 , which is intentionally accepted in exchange for a correction table with simple quantized values. A more finely quantized table can move the bias closer to zero, but doing so would increase correction-generation complexity. Therefore, the selected table represents a hardware-oriented compromise rather than the mathematically optimal conditional-mean estimator.

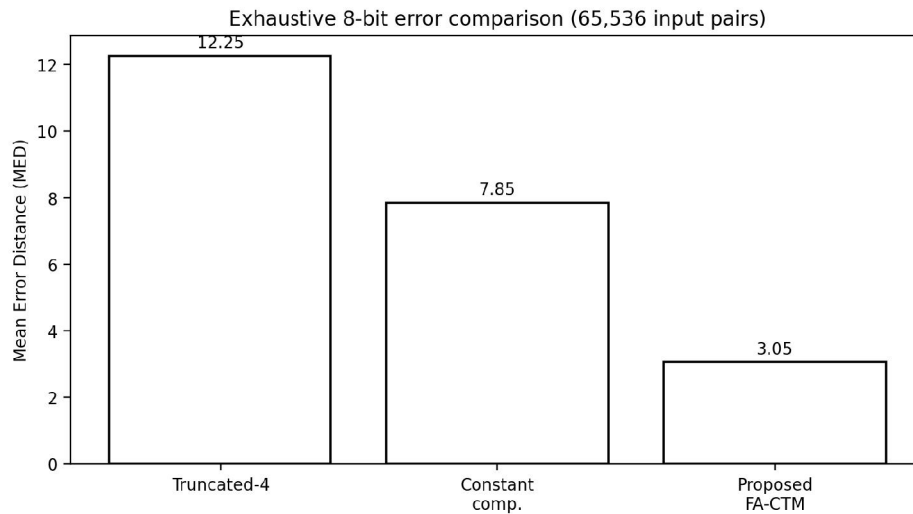


Fig. 3. Mean error distance for the approximate 8-bit designs.

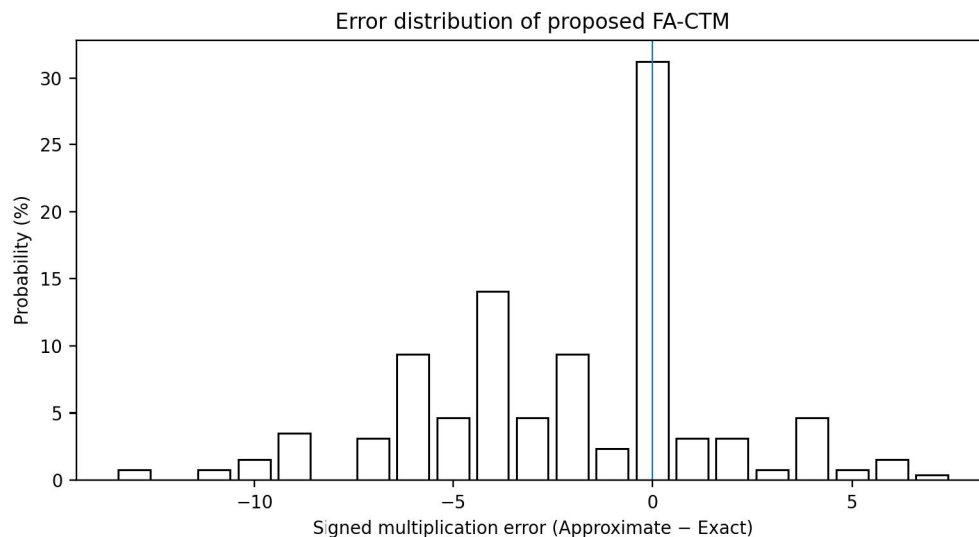


Fig. 4. Signed error distribution of the proposed 8-bit FA-CTM. The compact support (-13 to $+8$ in this experiment) limits worst-case numerical deviation.

B. Multiply-Accumulate Error Accumulation

Error statistics at the individual multiplication level do not fully describe MAC behavior because repeated biased errors can accumulate. For 10,000 random 64-term dot products, uncompensated truncation produced a mean absolute dot-product error of 784.52 and a mean relative error of 0.0763%. FA-CTM reduced these values to 135.94 and 0.0132%, respectively. The 99th-percentile relative error decreased from 0.1072% to 0.0216%. The mean signed dot-product error changed from -784.52 for truncation to -135.94 for FA-CTM, showing that frontier compensation substantially limits systematic drift across repeated multiplication.

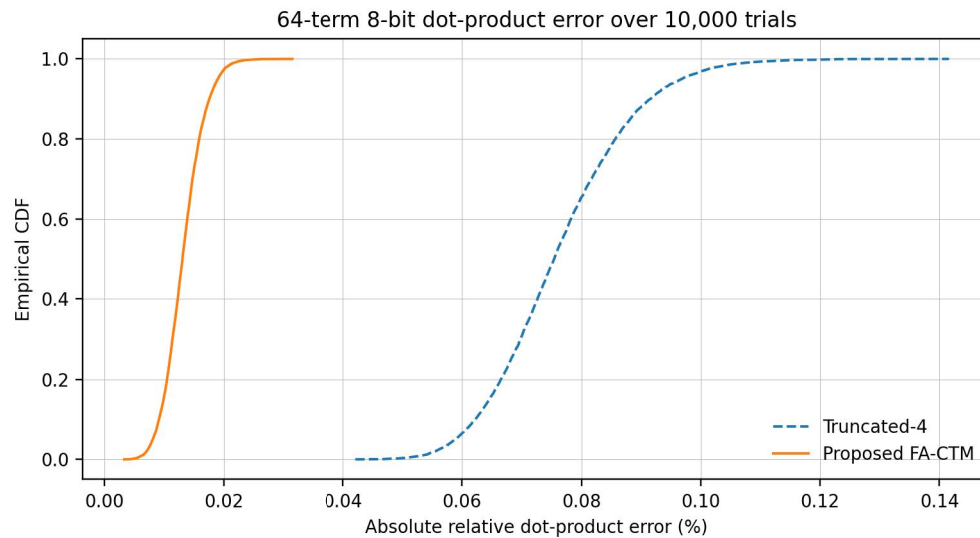


Fig. 5. Empirical CDF of relative error for 64-term 8-bit dot products.

C. Image-Domain Evaluation

In fixed-point brightness scaling, the numerical error of each multiplication is divided by 255 and rounded to the nearest pixel level. Consequently, low multiplication errors are often visually masked. Nevertheless, the proposed compensation provides a measurable improvement: the PSNR relative to the exact result increases from 61.56 dB for simple truncation to 67.45 dB for FA-CTM, while mean absolute pixel error decreases from 0.04538 to 0.01169 gray levels. This experiment is intentionally simple and reproducible; broader filtering, compression, and neural-network evaluation should be performed when the target application is selected.

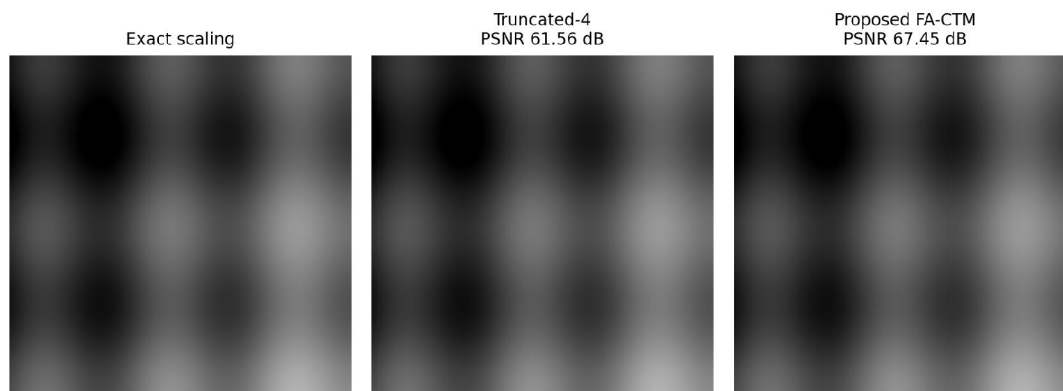


Fig. 6. Fixed-point image scaling benchmark. The proposed multiplier tracks the exact output more closely than uncompensated truncation.

D. 16-bit Scalability Check

Table III. Sampled 16-bit validation (500,000 random operand pairs, k=8)

Design	MED	NMED	MRED	MaxED	Mean signed error
Truncated-8	448.014	1.043e-07	9.238e-06	1793	-448.014
Proposed FA-CTM	68.066	1.585e-08	1.617e-06	321	1.399

At 16 bits, eight-column truncation yields a sampled MED of 448.014 and MaxED of 1793. The scaled FA-CTM estimator reduces MED to 68.066 and MaxED to 321, corresponding to an MED reduction of 84.8%. The remaining mean signed error is close to zero relative to the full 32-bit product range. These results suggest that the frontier count continues to carry useful information about the discarded triangular region as word length and truncation depth increase.

E. Comparison with the Literature

Direct numerical comparison of power, delay, or silicon area across approximate-multiplier papers is generally invalid unless the designs are synthesized under the same technology library, voltage, constraints, operand width, signedness, and error target. For example, the literature includes 28-nm, 40-nm, and 45-nm studies as well as FPGA-oriented evaluations [4], [9]–[14]. Therefore, Table IV compares design mechanisms rather than mixing technology-dependent values. The results generated in this paper are presented separately as exhaustive/sampled arithmetic measurements and partial-product-domain activity counts.

Table IV. Mechanism-level comparison

Technique	Approximation location	Bias handling	Configuration	Distinctive characteristic
Underdesigned block [3]	Small multiplier cells	Implicit	Design-time	Hierarchical inaccurate blocks
Partial error recovery [4]	Carry propagation	Configurable recovery	Configurable	Accuracy–power control
Approx. compressors [5],[11],[12],[14]	Reduction tree	Cell-dependent	Design-time	Strong PPA/error design space
DRUM [6]	Operand significant range	Explicitly targets unbiased error	Configurable	Dynamic-range selection
RoBA [8]	Operand rounding	Rounding dependent	Design-time	Simplifies multiply operation
FA-CTM (this work)	Discarded low columns	Frontier-conditioned correction	k and correction table	Small estimator around truncation boundary

VI. IMPLEMENTATION CONSIDERATIONS

The proposed correction network is naturally parameterized by k. In an RTL implementation, the main multiplier datapath should avoid generating the fully discarded region; simply computing an exact product and masking lower bits would preserve most of the original hardware and would not realize the intended savings. The frontier terms should be tapped directly from operand bits, counted by a compact combinational network, and decoded into an aligned correction word. For k=4, a two- or three-level population network is sufficient for four one-bit terms.

Timing closure depends on how the correction is merged with the retained product. One option is to inject C(f) into the lower active columns of the reduction tree so that compensation is absorbed before the final carry-propagate adder. Another option is a parallel correction adder. The first is likely preferable for high-throughput ASIC implementations

because it can exploit existing compressor stages, while the second is simpler for initial RTL verification and FPGA prototyping.

The correction table can also be workload-trained. This manuscript uses uniform operand statistics to avoid overfitting. In a neural accelerator, activation and weight distributions are rarely uniform, so $C(f)$ may be recalibrated from representative workload traces. Such calibration should be performed on training data only and then verified on held-out traces to ensure that the arithmetic block does not become application-specific in an uncontrolled manner.

For signed two's-complement multiplication, direct reuse of the unsigned truncation rule is not sufficient because sign-extension and negative partial products change the error distribution. A signed extension should therefore operate on a Booth-encoded or Baugh–Wooley representation and derive its frontier statistics from the signed partial-product structure. This extension is left as future work.

VII. LIMITATIONS AND VALIDITY

The principal limitation of the present study is that hardware results are architecture-level rather than technology-mapped. The reported 9.375% reduction in generated partial-product signals and the 9.35% reduction in partial-product toggle events do not equal complete-chip area or dynamic-power savings because the compensation logic and reduction network contribute additional cost. Post-synthesis area, slack, and power should be reported using a named standard-cell library and reproducible constraints before claiming a specific PPA improvement over published hardware. Second, the application experiments are intended to demonstrate error behavior, not to establish task-level AI accuracy. A complete edge-AI evaluation should insert the multiplier into an RTL or hardware-aware inference flow for a selected network and dataset. Finally, the 16-bit result is based on random sampling rather than exhaustive enumeration because the full input space contains more than four billion operand pairs.

VIII. CONCLUSION

This paper presented FA-CTM, a frontier-aware compensated truncated multiplier for error-tolerant VLSI datapaths. The design removes low-weight partial products from the main accumulation network while retaining only the highest discarded diagonal as a compact estimator of the omitted numerical contribution. A quantized correction table substantially reduces the systematic bias and error magnitude of conventional truncation. Exhaustive 8-bit evaluation demonstrated $MED=3.055$, $NMED=4.70 \times 10^{-5}$, $MRED=1.92 \times 10^{-3}$, and $MaxED=13$. Compared with simple four-column truncation, MED falls by 75.1% and MAC-level mean relative error falls from 0.0763% to 0.0132%. The proposed partial-product domain uses 58 instead of 64 generated terms and shows 9.35% fewer partial-product toggles under random input transitions. A 16-bit sampled experiment confirms that the frontier principle scales to a deeper truncation boundary.

Future work should implement the design in synthesizable SystemVerilog, map it to a common open or commercial standard-cell library, perform post-layout timing and power analysis, and evaluate signed multiplication and end-to-end neural inference. Those steps would convert the present architecture-level study into a fully technology-characterized VLSI implementation.

REFERENCES

- [1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in 18th IEEE European Test Symposium (ETS), 2013, pp. 1–6, doi: 10.1109/ETS.2013.6569370.
- [2] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate Computing: A Survey," IEEE Design & Test, vol. 33, no. 1, pp. 8–22, Feb. 2016, doi: 10.1109/MDAT.2015.2505723.
- [3] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," in 2011 24th International Conference on VLSI Design, 2011, pp. 346–351, doi: 10.1109/VLSID.2011.51.

- [4] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014, pp. 1–4, doi: 10.7873/DATE.2014.108.
- [5] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and Analysis of Approximate Compressors for Multiplication," IEEE Transactions on Computers, vol. 64, no. 4, pp. 984–994, Apr. 2015, doi: 10.1109/TC.2014.2308214.
- [6] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications," in 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015, pp. 418–425, doi: 10.1109/ICCAD.2015.7372600.
- [7] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation," IEEE Transactions on Computers, vol. 65, no. 8, pp. 2638–2644, Aug. 2016, doi: 10.1109/TC.2015.2493547.
- [8] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 2, pp. 393–401, Feb. 2017, doi: 10.1109/TVLSI.2016.2587696.
- [9] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing," IEEE Transactions on Computers, vol. 66, no. 8, pp. 1435–1441, Aug. 2017, doi: 10.1109/TC.2017.2672976.
- [10] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, "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, Sep. 2018, doi: 10.1109/JETCAS.2018.2832204.
- [11] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate Multipliers Based on New Approximate Compressors," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 12, pp. 4169–4182, Dec. 2018, doi: 10.1109/TCSI.2018.2839266.
- [12] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. Di Meo, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 3021–3034, Sep. 2020, doi: 10.1109/TCSI.2020.2988353.
- [13] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, and J. Han, "Scalable Construction of Approximate Multipliers With Formally Guaranteed Worst Case Error," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 11, pp. 2572–2576, Nov. 2018, doi: 10.1109/TVLSI.2018.2856362.
- [14] L. Sayadi, S. Timarchi, and A. 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, Apr. 2023, doi: 10.1109/TCSI.2023.3242558.
- [15] G. Rodrigues, F. L. Kastensmidt, and A. Bosio, "Survey on Approximate Computing and Its Intrinsic Fault Tolerance," Electronics, vol. 9, no. 4, Art. no. 557, Mar. 2020, doi: 10.3390/electronics9040557.
- [16] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, pp. 258–261, doi: 10.23919/DATE.2017.7926993