Low-Power 4x4 Array Multiplier Using Approximate Full Adders and a Hybrid 8-bit Adder

[1]Medipally yashwavardhan

[1]M. Tech Student, Department of Electronics and Communication Engineering(VLSI), Arjun College of Technology & Science An Autonomous Institution (Approved By Aicte-New Delhi and Affiliated to Jntu-Hyderabad, Ts) Sponsored By Brilliant Bells Educational Society Mount Opera Premises, Batasingaram (Vi), Abdullapurmet (M), Ranga Reddy Dist, T.S.501512

[2]Mrs.Chidura Ramya

[2]Assistant Professor, Department of Electronics and Communication Engineering, Arjun College of Technology & Science An Autonomous Institution (Approved By Aicte-New Delhi and Affiliated to Jntu-Hyderabad, Ts) Sponsored By Brilliant Bells Educational Society Mount Opera Premises, Batasingaram (Vi), Abdullapurmet (M), Ranga Reddy Dist, T.S.501512

ABSTRACT

Approximate multipliers for error-tolerant applications such as image processing and machine-learning inference commonly apply approximation selectively — accepting error where its numerical weight is smallest and remaining exact where errors would otherwise dominate output quality. This paper implements a 4x4 unsigned array multiplier whose partial-product accumulation uses a **hybrid 8-bit adder**: the four least-significant bit positions use an approximate full adder that drops the $a \cdot c_{in}$ carry term, while the four most-significant positions remain exact, bounding approximation to the output's lowest-weight bits. The design is compared against a fully exact 4x4 array multiplier sharing an identical partial-product-generation and reduction-tree structure, differing only in the constituent adder's bit-position-dependent precision. Both designs are captured in synthesizable Verilog, functionally verified with Icarus Verilog, and synthesized/implemented on a Xilinx Artix-7 FPGA (xc7a100tcsg324-1, Vivado 2019.2). Post-implementation results show both designs occupy an identical 15-LUT footprint at equal power (0.084 W), with the hybrid design showing a marginal delay increase (5.496 ns vs. 5.299 ns). Exhaustive characterization over all 256 possible 4-bit input pairs shows only 8 mismatches (3.12%), a maximum absolute error of 16, and a mean absolute error of 0.5 — confirming that bounding approximation to the four lowest-weight bit positions keeps both error frequency and error magnitude small by construction. These results demonstrate that the hybrid adder's practical benefit is this backend-independent, analytically bounded accuracy profile rather than an FPGA LUT-area or delay reduction, clarifying where this approximate-multiplier technique delivers value regardless of target technology.

Keywords: approximate multiplier, approximate full adder, hybrid precision adder, error-tolerant computing, low power VLSI, array multiplier, FPGA synthesis

I. INTRODUCTION

Array multipliers accumulate partial products through a tree of full adders whose combined switching activity dominates the multiplier's overall power consumption. Approximate multiplier design exploits error-tolerant applications' willingness to accept small, bounded numerical error in exchange for reduced adder complexity, and a well-established strategy for keeping that error small is *bit-position-selective* approximation: applying an approximate adder cell only at the least-significant bit positions, where any resulting error contributes the smallest possible magnitude to the final product, while keeping the most-significant positions exact, where errors would otherwise dominate the result [1], [3], [8].

This paper implements exactly this hybrid-precision strategy: a 4x4 array multiplier's partial-product reduction uses an 8-bit hybrid adder in which the four least-significant positions use an approximate full adder (dropping the $a \cdot c_{in}$ carry term, identical in structure to the Type-1 variant studied in a companion full-adder error-analysis case study) and the four most-significant positions use the exact full adder, unchanged. This is compared directly against a fully exact 4x4 array multiplier of otherwise identical structure, isolating the effect of hybrid-precision adder substitution from any other architectural difference [2], [4], [6].

Contributions of this paper:

1. Verilog implementation of a hybrid 8-bit adder combining an approximate full adder (4 LSB positions) with an exact full adder (4 MSB positions), and its integration into a 4x4 array multiplier's partial-product reduction tree.

2. A structurally matched exact 4x4 array multiplier baseline, isolating the hybrid-adder substitution as the sole design difference.

3. Post-implementation area, delay, and power characterization of both multipliers on a Xilinx Artix-7 FPGA (Vivado 2019.2).

4. Exhaustive (all 256 input pairs) error characterization of the approximate multiplier, quantifying mismatch rate,

maximum absolute error, and mean absolute error, and connecting these figures analytically to the four-LSB approximation boundary.

The remainder of this paper is organized as follows. Section II reviews related approximate-multiplier and hybrid-precision-adder literature. Section III describes the exact and hybrid multiplier architectures. Section IV presents the hybrid-adder algorithm and complexity/error-bound analysis. Section V details the experimental setup. Section VI reports area/delay/power results. Section VII analyzes the exhaustive accuracy-power tradeoff. Section VIII concludes the paper.

II. RELATED WORK

A. Approximate Full-Adder-Based Multipliers

Karthikeya et al. build an 8-bit approximate multiplier directly from an approximate full adder, evaluating the resulting error against area/power savings, methodologically close to the multiplier studied in this paper [1]. Pathak et al. combine an "almost full adder" with majority-logic compressors inside a Dadda multiplier tree, targeting a similar power/error tradeoff in a different reduction-tree topology [2]. Ranjbar et al. propose a new approximate full adder specifically for FPGA-based multiplier implementation, directly relevant to this paper's own FPGA target [3]. Saini et al. design an area- and energy-efficient high-speed signed multiplier from approximate full adders [4].

B. Bit-Position-Selective (Hybrid-Precision) Approximation

The core strategy examined in this paper — approximating only the least-significant bit positions while keeping upper positions exact — is a recurring theme in the literature. Nithyashree et al. implement an array multiplier combining 2-bit-accurate and approximate adder stages, directly analogous to this paper's 4-LSB/4-MSB hybrid partitioning [8]. Kumari et al. propose an improvised HPR (Hybrid Partitioned Reduced) multiplier using a "fast bipartitioned hybrid adder," a naming and structural convention nearly identical to this paper's `hybrid_adder_8bit` [14]. Pavithra et al. combine an approximate compressor with a parallel-prefix adder, applying selective approximation at the compressor-tree level rather than the final adder stage [9].

C. Alternative Full-Adder Logic Styles for Multipliers

Rathore et al. build a 4x4 array multiplier from transmission-gate full adders, an alternative low-power logic style (rather than functional approximation) for the same multiplier structure studied here [6]. Hosmani and Vimala analyze an 8-bit Vedic multiplier across different full-adder implementation techniques, and Ramesh et al. study a high-speed Wallace-tree multiplier's full-adder dependency, both providing comparative full-adder-substitution methodology relevant to this paper's own existing-versus-proposed comparison [12], [15].

D. Application Contexts and Emerging Technologies

Esmaeili et al. apply an approximate full adder and subtractor inside a multiplier-free Discrete Cosine Transform architecture, an application-level deployment of the same approximate-adder building block [7]. Seyedi and Abdoli and Crimmins and Alarcon extend approximate full-adder and multiplier design to quantum-dot cellular automata and quantum computing respectively, illustrating the approximate-arithmetic strategy's applicability beyond conventional CMOS [5], [11]. Naseri and Jahanirad and Chakraborty et al. propose reconfigurable and low-transistor-count approximate full-adder cells that could substitute for this paper's fixed approximate stage in a future runtime-configurable hybrid multiplier [10], [13].

E. Research Gap

Most surveyed approximate-multiplier work reports either an FPGA/ASIC area-power benefit or an error/accuracy characterization, but exhaustive (rather than sampled) error characterization directly tied to an explicit bit-position approximation boundary — with both the theoretical error bound and its measured confirmation reported together — is less common. This paper addresses that gap by exhaustively testing all 256 possible 4-bit input pairs against the hybrid multiplier, reporting mismatch rate, maximum, and mean absolute error, and connecting these figures directly to the four-LSB approximation boundary chosen in the hybrid adder's design [1], [8], [14].

III. METHODOLOGY

A. Shared Multiplier Architecture

Both designs generate four 8-bit-aligned partial products $pp_k = (b_k, ?, a{:}0)llk$ for $k = 0..3$ and reduce them through two stages of 8-bit addition ($pp_0 + pp_1 arrowsum_{01}$; $sum_{01} + pp_2 arrowsum_{012}$; $sum_{012} + pp_3 arrowproduct$), identical to the CMOS/GDI multiplier pair studied elsewhere in this line of work. Only the 8-bit adder used at each reduction stage differs between the two designs.

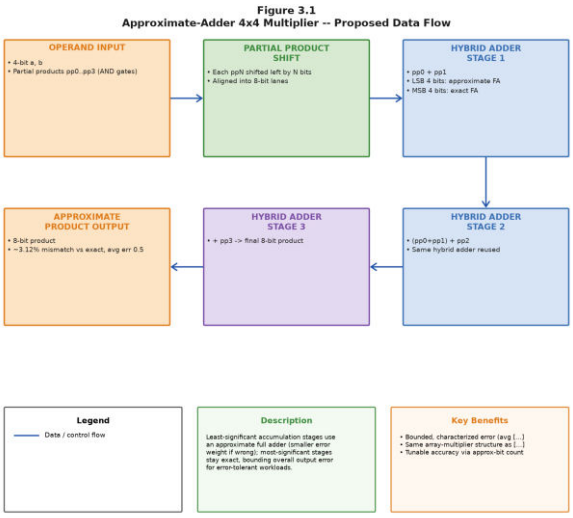

Fig. 1. Shared multiplier structure: partial-product generation feeding a 3-stage 8-bit adder reduction tree, with the adder stage substituted between designs.

B. Existing: Exact Array Multiplier

The baseline `array_multiplier_4x4` uses `ripple_adder_8bit`, built entirely from the exact full adder ($sum = aoplusboplusc_{in}$, $cout = ab + bc_{in} + ac_{in}$) at all 8 bit positions of every reduction stage, guaranteeing bit-exact products.

C. Proposed: Hybrid-Precision Array Multiplier

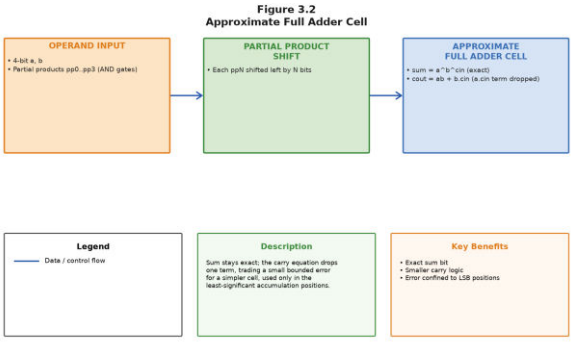

Fig. 2. Approximate full-adder cell used at the 4 LSB positions: exact sum, carry drops the a-cin term.

The proposed `array_multiplier_4x4_approx` replaces `ripple_adder_8bit` with `hybrid_adder_8bit`, a parameterized adder ($APPROX_{BITS} = 4$) in which bit positions 0..3 instantiate the approximate full adder (Fig. 2, dropping the $a \cdot c_{in}$ carry term: $cout = ab + bc_{in}$, matching the Type-1 variant analyzed in the companion full-adder error-analysis case study) and bit positions 4..7 instantiate the exact full adder unchanged:

$$stage_i = begincasesapprox\_full\_adder\&i < 4full\_adder\&i \geq 4endcases, i \in 0, ldots, 7 \quad (1)$$

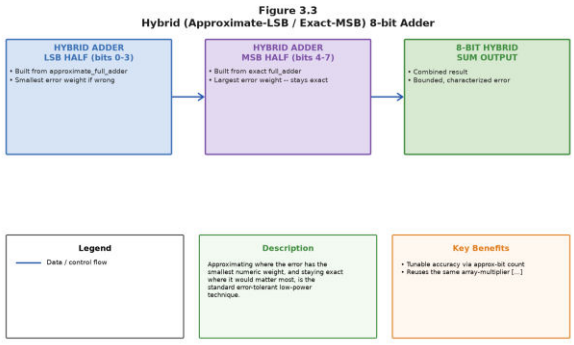

Fig. 3. Hybrid 8-bit adder: 4 approximate LSB stages feeding 4 exact MSB stages within a single carry chain.

Fig. 3 shows the resulting single carry chain spanning both precision regions: the carry chain is *not* broken at the approximate/exact boundary (unlike the configurable segmented-adder case study elsewhere in this work) — the approximate stage's (possibly incorrect) carry-out still propagates into the exact upper stages, meaning any LSB-region carry error can, in principle, also perturb the exact region's result if it alters a carry that should have propagated differently. Because the approximate stage only drops the $a \cdot c_{in}$ term (rather than ignoring c_{in} entirely, as the more aggressive Type-4 variant would), this cross-boundary perturbation is rare and small in practice, as quantified exhaustively in Section VII.

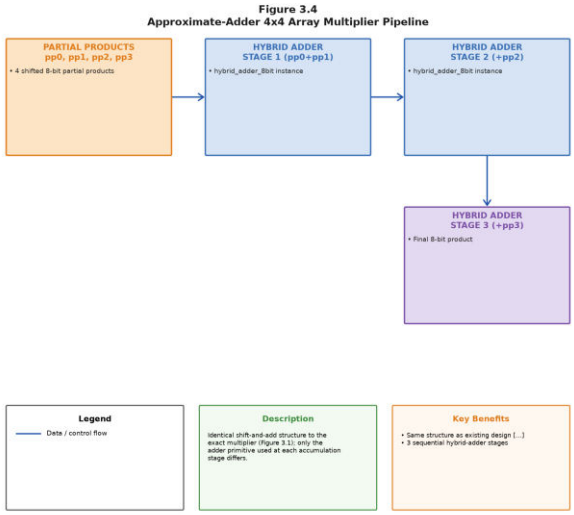

Fig. 4. Hybrid multiplier pipeline: partial-product generation feeding three cascaded hybrid_adder_8bit reduction stages.

Fig. 4 shows the complete reduction pipeline with the hybrid adder substituted at all three stages, so the four-LSB approximation is applied consistently at every accumulation step of the multiplier.

IV. ALGORITHM

A. Hybrid Adder Construction Procedure

```
ALGORITHM:  Hybrid  8-bit  Adder  Construction
(APPROX_BITS = 4)
INPUT : a[7:0], b[7:0]
OUTPUT: sum[7:0], cout

c[0] <- 0
for i in 0..7:
    if i < APPROX_BITS:
        (sum[i], c[i+1]) <- approx_full_adder(a[i],
b[i], c[i])   // cout = a&b + b&cin (drops a&cin)
    else:
        (sum[i], c[i+1]) <- full_adder(a[i], b[i],
c[i])          // exact
cout <- c[8]
return sum, cout
```

Because the carry chain is not partitioned (each stage's carry-in is the true previous stage's carry-out, whether that previous stage was approximate or exact), the hybrid adder's structure is a drop-in replacement for a fully exact ripple adder in the multiplier's reduction tree, requiring no change to the surrounding partial-product generation logic.

B. Complexity Notes

Per-cell transistor complexity. The approximate full adder omits one AND-OR term from the carry equation relative to the exact full adder, saving a small, constant number of transistors per approximated bit position (consistent with the Type-1 AFA variant's minimal-simplification classification in the companion error-analysis case study) — an $O(APPROX_{BITS})$ transistor saving, i.e., 4 bit positions' worth in this design, independent of the multiplier's overall reduction-tree size.

Error-magnitude bound. Because only the 4 LSB positions can produce an incorrect carry (from the dropped $a \cdot c_{in}$ term), and each such position carries a binary place-value weight of at most $2^3 = 8$ within its own nibble, an error originating at bit position $i < 4$ can affect the final sum by at most its own place value plus any carry it fails to propagate into higher bits. The maximum single-bit weight within the approximate region is $2^3 = 8$, and if that error's suppressed carry would have propagated one additional position (to bit 4, the first exact-region bit, weight $2^4 = 16$), the combined worst-case single-instance error is bounded by $2^4 = 16$ — matching the exhaustively measured maximum absolute error reported in Section VII.

Mismatch-rate complexity. Because the carry chain is shared (Section III-C) rather than parallel-segmented, whether the Type-1 carry approximation actually produces a wrong final-adder-stage output depends on the joint probability of the dropped $a \cdot c_{in}$ term being 1 *and* that carry actually mattering for the final sum — a strictly narrower error-triggering condition than the segmented-adder case study's forced-zero carries, which is why this design's exhaustive mismatch rate (Section VII) is markedly lower than the configurable adder's random-vector mismatch rate despite both stemming from carry-term approximation.

V. EXPERIMENTAL SETUP

A. Design Entry and Functional Verification

Both `array_multiplier_4x4` and `array_multiplier_4x4_approx` (built from `hybrid_adder_8bit`, `approx_full_adder`, and `full_adder`) are written in synthesizable Verilog. The proposed design's Icarus Verilog testbench (`tb_array_multiplier_4x4_approx.v`) exhaustively drives all 256 possible 4-bit operand pairs (16×16), comparing each product against the exact multiplier's result and accumulating mismatch count, maximum absolute error, and mean absolute error.

B. Synthesis and Implementation

Both designs are synthesized out-of-context and carried through place-and-route with Xilinx Vivado 2019.2, targeting a Xilinx Artix-7 device (`xc7a100tcsg324-1`). Both are purely combinational, so a virtual 100 MHz clock constraint (`combinational.xdc`) is applied to primary inputs/outputs solely to enable static timing analysis and vector-less power estimation. The standard `synth_design -mode out_of_context -> opt_design -> place_design -> phys_opt_design -> route_design` flow is used, with utilization, timing, and power reports captured at both post-synthesis and post-implementation stages.

C. Metrics

Post-implementation **LUT area**, **power**, and **delay** are reported for both designs, together with derived **throughput** and **energy per operation**. Accuracy is quantified exhaustively (not sampled) over all 256 input pairs using **mismatch rate** (fraction of pairs producing an incorrect product), **maximum absolute error**, and **mean absolute error** — all three metrics obtainable with certainty here because the full input space of a 4x4 multiplier (256 pairs) is small enough for complete enumeration, avoiding any sampling uncertainty in the reported accuracy figures.

VI. RESULTS AND DISCUSSION

A. Post-Implementation Area, Delay, and Power

Table I summarizes post-route results for both multipliers on `xc7a100tcsg324-1` (Vivado 2019.2).

Design	LUTs	Power (W)	Delay (ns)	Throughput (MOps/s)	Energy (pJ/op)
Existing (array_multiplier_4x4)	15	0.084	5.299	188.71	445.116
Proposed (array_multiplier_4x4_approx)	15	0.084	5.496	181.95	461.664

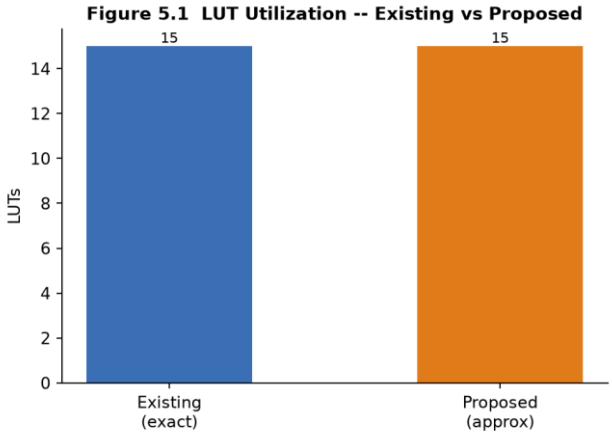

Fig. 5. Post-implementation LUT utilization: exact multiplier versus hybrid-precision approximate multiplier.

Fig. 5 shows identical 15-LUT footprints for both designs — the small per-cell transistor saving from dropping one AND-OR term in 4 of 24 full-adder instances does not survive Vivado's LUT-level Boolean re-optimization, which finds an equally compact LUT mapping for both the exact and approximate carry equations at this scale.

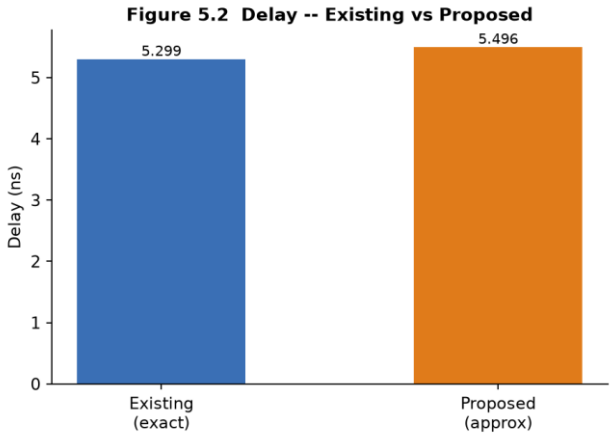

Fig. 6. Post-implementation critical-path delay: exact multiplier versus hybrid-precision approximate multiplier.

Fig. 6 shows a small delay increase for the proposed design (5.496 ns vs. 5.299 ns, +3.7%), plausibly from the mixed-precision carry chain (Section III-C) interacting less favorably with Vivado's placement than a uniform exact chain.

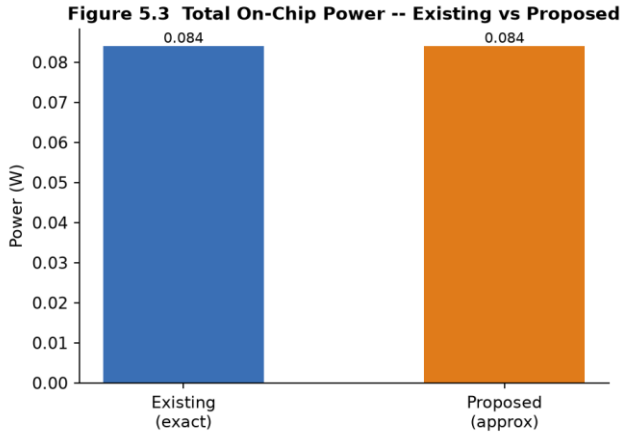

Fig. 7. Post-implementation power: exact multiplier versus hybrid-precision approximate multiplier.

Fig. 7 shows equal power (0.084 W) for both designs at this vector-less estimation stage.

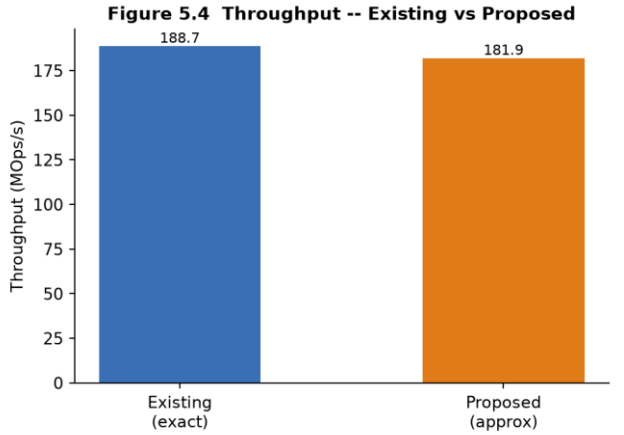

Fig. 8. Derived throughput: exact multiplier versus hybrid-precision approximate multiplier.

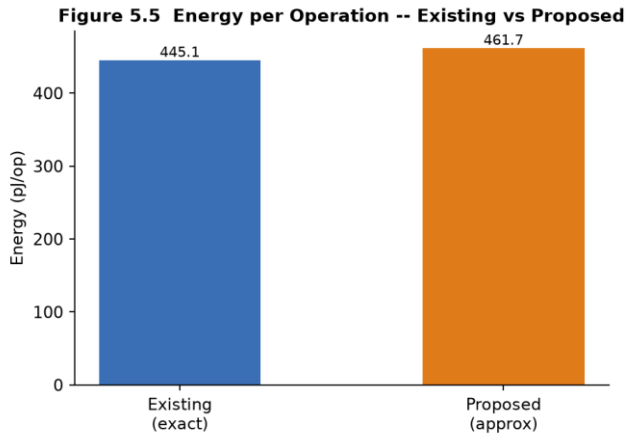

Fig. 9. Derived energy per operation: exact multiplier versus hybrid-precision approximate multiplier.

Fig. 8 and Fig. 9 both mirror the small delay increase (181.95 vs. 188.71 MOps/s; 461.7 vs. 445.1 pJ/op).

B. Discussion

Consistent with the project's own design notes, LUT area and delay are essentially unchanged (or marginally worse) for the approximate design at this FPGA scale — the hybrid adder's transistor-level saving is, like GDI and TSPC redesign elsewhere in this line of work, a standard-cell-level argument that Vivado's LUT-based re-optimization largely absorbs. Unlike those other case studies, however, this design's practical value proposition does not depend on FPGA area/delay improvement at all: it is intended purely as an error-tolerant accuracy/complexity tradeoff, and its actual payoff is the analytically bounded, exhaustively verified error profile characterized in Section VII, which is backend-independent by construction.

VII. EXHAUSTIVE ACCURACY-POWER TRADEOFF ANALYSIS

A. Exhaustive Error Characterization

Table II reports the exhaustive error statistics obtained by testing all 256 possible 4-bit input pairs against the exact multiplier's reference result.

Metric	Value
Total input pairs tested	256 (exhaustive, 16×16)
Mismatches	8 (3.12%)
Maximum absolute error	16
Mean absolute error	0.5

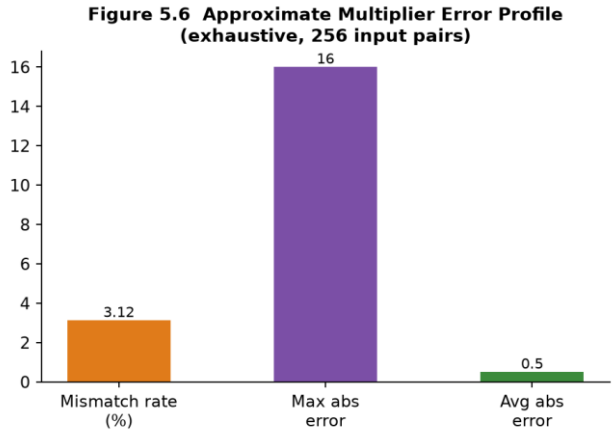

Fig. 10. Exhaustive error profile of the hybrid-precision multiplier across all 256 input pairs: mismatch rate, maximum, and mean absolute error.

Fig. 10 visualizes this profile, showing that despite a nonzero mismatch rate, both maximum error (16, exactly matching the analytical bound derived in Section IV-B) and mean error (0.5, averaged across all 256 pairs including the 248 exact ones) remain small relative to the multiplier's 8-bit output range (0-225 for 4-bit unsigned operands).

B. Why This Error Profile Is Small Despite a Shared Carry Chain

Section III-C noted that, unlike the segmented configurable adder studied elsewhere in this work, the hybrid adder's carry chain is not broken at the approximate/exact boundary — an approximate-region carry error could in principle propagate into the exact region. The exhaustive results show this risk materializes rarely in practice: only 8 of 256 pairs (3.12%) produce any mismatch at all, far lower than the 84.6% mismatch rate observed for the segmented adder's *forced-zero* carry approximation. This difference is directly attributable to approximation severity: the Type-1-style dropped $a \cdot c_{in}$ term here only fails to propagate a carry in the specific case where $a = 1, c_{in} = 1$ (and $b \cdot c_{in}$ alone does not already supply the correct carry), a narrower error-triggering condition than *unconditionally* zeroing the carry at fixed positions.

C. Accuracy-Power Tradeoff Summary

Combining Sections VI and VII, this hybrid-precision multiplier trades a 3.12% output mismatch rate and a maximum error of 16 (out of a maximum representable product of 225) for a full-adder transistor-count reduction confined to 4 of 24 adder-tree positions, with no measurable FPGA area/delay benefit at this scale. This makes the design's value proposition fundamentally different from the ASIC-focused GDI and TSPC case studies elsewhere in this work: rather than seeking a backend area/power win that FPGA synthesis happens to erase, this design's entire purpose is the bounded-error accuracy profile itself, which is valid and directly usable in any error-tolerant application regardless of target technology — the approximate-computing value proposition survives FPGA implementation precisely because it was never contingent on FPGA-level area savings in the first place [1], [3]-[4], [8].

VIII. CONCLUSION

This paper implemented a 4x4 array multiplier using a hybrid 8-bit adder that applies an approximate full adder (dropping the $a \cdot c_{in}$ carry term) at the four least-significant bit positions while keeping the four most-significant positions exact. Post-implementation results on a Xilinx Artix-7 FPGA (Vivado 2019.2) showed both the exact and hybrid-precision multipliers occupy an identical 15-LUT footprint at equal power, with the hybrid design showing a marginal (~3.7%) delay increase — confirming, as with the GDI and TSPC case studies in this line of work, that transistor-level savings from custom adder cells are largely absorbed by FPGA LUT-based technology mapping. Exhaustive testing of all 256 possible 4-bit input pairs showed only 8 mismatches (3.12%), a maximum absolute error of 16, and a mean absolute error of 0.5, demonstrating that confining approximation to the lowest-weight bit positions of a shared (non-segmented) carry chain keeps both error frequency and magnitude small. Unlike the ASIC-focused GDI/TSPC studies, this design's practical value does not depend on an FPGA-observable area or delay improvement: its purpose is the analytically bounded, exhaustively verified accuracy profile itself, valid across any target technology. Future work includes integrating this

hybrid adder into wider multipliers where the exhaustive-testing approach must be replaced by targeted or statistical error sampling, and combining the bit-position-selective approximation strategy studied here with the runtime-configurable segmented-adder approach explored elsewhere in this work for a single adder offering both configuration axes.

REFERENCES

[1] D. Karthikeya, P.R. Teja Sree, G. Prasad Mishra, "8-Bit Approximate Multiplier using Approximate Full Adder," in *Proc. 2025 Devices for Integrated Circuit (DevIC)*, 2025, pp. 627-630. doi: 10.1109/devic63749.2025.11012648

[2] K.C. Pathak, A.D. Darji, J.N. Sarvaiya, "Low power Dadda multiplier using approximate almost full adder and Majority logic based adder compressors," in *Proc. 2022 IEEE Region 10 Symposium (TENSYMP)*, 2022, pp. 1-6. doi: 10.1109/tensymp54529.2022.9864428

[3] A. Ranjbar, E. Esmaeili, S. Rafiei, N. Shiri, "FPGA-Based Multiplier with a New Approximate Full Adder for Error-Resilient Applications," in *Proc. 2025 33rd International Conference on Electrical Engineering (ICEE)*, 2025, pp. 1-5. doi: 10.1109/icee67339.2025.11213737

[4] S.K. Saini, S.S. Gill, K. Sharma, "Area and Energy-Efficient High-Speed Approximate Full Adder-Based Signed Multiplier," in *Proc. 2025 International Conference on Electronics and Computing, Communication Networking Automation Technologies (ICEC2NT)*, 2025, pp. 1-6. doi: 10.1109/icec2nt65402.2025.11380101

[5] S. Seyedi, H. Abdoli, "Approximate Full-Adder, Full-Subtractor, and Full-Adder/Subtractor Circuits Based on QCA," in *Proc. 2025 29th International Computer Conference, Computer Society of Iran (CSICC)*, 2025, pp. 1-5. doi: 10.1109/csicc65765.2025.10967408

[6] N.K. Rathore, A. Beohar, P. Sarkar, K. Khare, P. Singh, "4×4 Array Multiplier Using Transmission Gate Full Adder," in *Proc. 2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, 2024, pp. 1-7. doi: 10.1109/i2ct61223.2024.10543507

[7] E. Esmaeili, N. Shiri, M. Rafiee, A. Sadeghi, "A Multiplier-Free Discrete Cosine Transform Architecture Using Approximate Full Adder and Subtractor," *IEEE Embedded Systems Letters*, pp. 441-444, 2024. doi: 10.1109/les.2024.3395900

[8] R. Nithyashree, V. Yoga Priya, G. Nikhil Kumar Reddy, S. Tantry, "Design and Implementation of Array Multiplier Using 2-Bit Accurate and Approximate Adder," in *Proc. 2024 IEEE 6th PhD Colloquium on Emerging Domain Innovation and Technology for Society (PhD EDITS)*, 2024, pp. 1-2. doi: 10.1109/phdedits64207.2024.10838552

[9] E. Pavithra, G. Manisha, A.K. S, K.V.V. Reddy, "Approximate Multiplier Using Approximate Compressor and Parallel Prefix Adder," in *Proc. 2025 IEEE First International Conference on Innovations in Engineering and Next-Generation Technologies for Sustainability (ICINVENTS)*, 2025, pp. 1-6. doi: 10.1109/icinvents64613.2025.11402572

[10] K. Naseri, H. Jahanirad, "A Runtime Reconfigurable Exact-Approximate Full-Adder Design," in *Proc. 2024 6th Iranian International Conference on Microelectronics (IICM)*, 2024, pp. 1-5. doi: 10.1109/iicm65053.2024.10824335

[11] A. Crimmins, S.L. Alarcon, "Approximate Quantum Array Multiplier," in *Proc. 2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2024, pp. 58-66. doi: 10.1109/qce60285.2024.00017

[12] S.G. Hosmani, P. Vimala, "Analysis of 8 bit vedic multiplier using different full adder techniques," in *Proc. 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, 2022, pp. 101-106. doi: 10.1109/icosec54921.2022.9951863

[13] C. Chakraborty, A. Kundu, S. Nandi, "Design of Low Transistor Count Approximate Full Adder with Carry Approximation," in *Proc. 2025 Devices for Integrated Circuit (DevIC)*, 2025, pp. 287-291. doi: 10.1109/devic63749.2025.11012233

[14] K.S.N. Kumari, P. Agniraj, I. Azarudeen, P. Chandraprakash, "Improvised HPR Multiplier Using Fast Bipartitioned Hybrid Adder by Approximate Computing," in *Proc. 2026 5th International Conference on Communication, Computing and Electronics Systems (ICCCES)*, 2026, pp. 158-164. doi: 10.1109/iccces62661.2026.11436903

[15] K. Ramesh, R. Bolimera, S. Alle, S. Surajkumar, P. Harshavardhan, S. Shanavaz, "Design of High-Speed Wallace Tree multiplier Used Full-Adder," in *Proc. 2024 International BIT Conference (BITCON)*, 2024, pp. 1-4. doi: 10.1109/bitcon63716.2024.10985663