International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

A Runtime-Configurable Exact/Approximate 16-bit Adder for Dynamic Accuracy-Power Tradeoffs

[1]Kyasaarapu Chaitanya

[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.Bheemagari Sandhya Rani

[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 adders typically fix their accuracy-power operating point at design time, forcing designers to choose a single tradeoff for the lifetime of the circuit even though workload accuracy requirements often vary at runtime. This paper presents a runtime-configurable 16-bit adder built from four cascaded 4-bit segment adders, in which a single control bit, `mode_approx`, selects between two operating modes using identical hardware: when `mode_approx = 0`, segment carries are chained normally, reproducing an exact ripple-carry result; when `mode_approx = 1`, each segment's carry-in is forced to zero, allowing all four segments to add in parallel at the cost of a bounded inter-segment carry-propagation error. The design is compared against a conventional exact 16-bit ripple-carry adder, both 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 the configurable adder matches the plain ripple adder's 16-LUT footprint while reducing critical-path delay from 8.623 ns to 7.599 ns (11.9% faster) and improving throughput from 115.97 to 131.6 MOps/s, even under the conservative worst-case timing view that covers both operating modes. Exhaustive and random-vector characterization of the approximate mode confirms an 84.6% mismatch rate against the exact result with a maximum absolute error of 4368 — exactly matching the theoretical worst case of $16 + 256 + 4096$ from the three inter-segment carry chains — validating that the observed error behavior is a direct, predictable consequence of the forced carry-in design rather than an artifact of implementation. These results demonstrate that a single reconfigurable adder can deliver both an exact reference mode and a materially faster approximate mode from one synthesized netlist, without any area penalty relative to a fixed-precision baseline.

Keywords: configurable approximate adder, runtime accuracy-power tradeoff, segmented adder, approximate computing, low-power VLSI, FPGA synthesis, carry propagation error

I. INTRODUCTION

Approximate arithmetic circuits typically fix a single accuracy-power operating point at design time: once an approximate adder is synthesized, its error behavior is baked into the netlist and cannot be relaxed back to exact operation if a later workload phase demands full precision. This is a significant limitation for real systems, where accuracy requirements often vary across execution phases — for example, an image pipeline may tolerate approximate addition during preview rendering but require exact addition for a final export pass, or a machine-learning accelerator may tolerate approximate accumulation during early training epochs but require exact accumulation near convergence.

Runtime-configurable approximate adders address this by embedding both an exact and an approximate datapath in the same synthesized hardware, selected by a control signal, so a single netlist serves both operating regimes without re-synthesis or reconfiguration [1], [3]-[4]. Segmented adder architectures are a natural fit for this approach: splitting a wide adder into narrower segments and controlling whether inter-segment carries propagate or are forced to a fixed value gives a low-cost, single-bit-controlled switch between exact chained operation and a faster, lower-power parallel-segment approximation [2], [6], [8].

This paper implements exactly this approach for a 16-bit adder: four 4-bit segment adders are cascaded, and a

`mode_approx` control bit either allows normal inter-segment carry propagation (exact mode) or forces every segment's carry-in to zero, enabling all four segments to add in parallel at the cost of a bounded, predictable carry-propagation error (approximate mode). The design is compared directly against a conventional 16-bit ripple-carry adder to quantify both the mode-selection overhead and the runtime power/timing benefit.

Contributions of this paper:

1. Verilog implementation of a runtime-configurable 16-bit adder using four 4-bit segments with a single-bit `mode_approx` control selecting exact chained-carry or approximate forced-carry-in operation.

2. Exhaustive and random-vector error characterization of the approximate mode, confirming the observed 84.6% mismatch rate and maximum absolute error of 4368 exactly match the theoretical worst case derivable from the three forced inter-segment carries.

3. Post-implementation area, delay, power, and throughput comparison against a conventional exact 16-bit ripple-carry adder on a Xilinx Artix-7 FPGA (Vivado 2019.2), under the conservative worst-case timing view spanning both operating modes.

4. A quantitative accuracy-power tradeoff analysis showing the configurable design's approximate mode is not merely "faster but wrong" but exhibits a bounded, analytically predictable error magnitude directly tied to segment boundary positions.

The remainder of this paper is organized as follows. Section II reviews related work on configurable and reconfigurable approximate adders. Section III describes the segmented architecture and mode-selection logic. Section IV presents the operating algorithm and complexity analysis. Section V details the experimental setup. Section VI reports area/delay/power results. Section VII analyzes the accuracy-power tradeoff across both modes. Section VIII concludes the paper.

II. RELATED WORK

A. Reconfigurable Exact/Approximate Adders

Oghostinos et al. integrate a configurable approximate adder directly into a single-cycle MIPS processor datapath, switching accuracy at the instruction level [1]. Naseri and Jahanirad propose a runtime-reconfigurable exact-approximate full-adder cell that can be composed into wider reconfigurable adders such as the one studied here [3]. Vendhan et al. present a reconfigurable-accuracy approximate adder with an explicit accuracy-selection mechanism evaluated across multiple operating points [4]. Pindoo et al. extend reconfigurability to a carry-look-ahead structure, introducing partitioned precision control with runtime-selectable accuracy across adder partitions — a segmentation strategy closely related to the four-segment design examined in this paper [2].

B. Segmented and Partitioned Approximate Adder Structures

Segmentation is a recurring strategy for bounding approximate-adder error to predictable regions of the output. I.K. and Francis compare a radix-4 adder against an accuracy-configurable radix-4 variant, showing how radix-level segmentation trades area for configurable accuracy [8]. R. R. et al. combine truncation with a Kogge-Stone carry structure, effectively segmenting the adder into an exact high-order region and an approximate low-order region [7]. Ghavami et al. propose a majority-based approximate adder specifically targeting FPGA fabrics, directly relevant to this paper's FPGA-based implementation and evaluation [12].

C. Error-Bounded and Application-Aware Approximate Adders

Several works aim to keep approximate-adder error analytically bounded or application-aware rather than unconstrained. Prihatiningrum et al. propose an area-efficient, error-reduced CMOS approximate adder explicitly designed to minimize worst-case error magnitude [5]. Seok et al. use single-input-pair-based computation for an efficient approximate adder with a constrained error profile [6]. Lopes et al. propose an approximate recombination line inside a 4-2 adder compressor, targeting compressor-tree approximate error containment relevant to approximate multiplier design [10]. J. V. et al. and Park and Ko target application-specific approximate adders for image processing and DNN inference in compute-in-memory arrays respectively, both contexts where a bounded, predictable error profile is essential for acceptable output quality [11], [13].

D. Comparative and Survey-Style Evaluations

Chakraborty et al. present a low-transistor-count approximate full adder with carry approximation, a cell-level building block that could substitute for this paper's segment adders in future hybrid designs [9]. Parameswari and Ananthalakshmi analyze a pruned, majority-logic-based adder from an area/error tradeoff perspective [14]. Nithyashree et al. provide a comparative analysis of multiple approximate adder variants within a ripple-carry-adder context, methodologically close to this paper's own existing-versus-proposed comparison structure [15].

E. Research Gap

Most surveyed configurable/reconfigurable adder designs report either area/power savings or error characterization, but rarely both from the same synthesized hardware evaluated end-to-end from RTL through FPGA place-and-route, and few explicitly verify that the observed error magnitude matches an analytically derived theoretical bound. This paper closes that gap by implementing a single-bit-controlled configurable 16-bit adder, characterizing its approximate-mode error exhaustively and via random-

vector sampling, and confirming the observed maximum absolute error matches the theoretical worst case derived directly from the three forced inter-segment carry positions — connecting the reconfigurable-adder literature [1]-[4] with the error-bound literature [5]-[6], [10] under one synthesizable, FPGA-verified design.

III. METHODOLOGY

A. Existing: Exact 16-bit Ripple-Carry Adder

The baseline is a conventional 16-bit ripple-carry adder built from 16 cascaded full-adder cells, with carry propagating serially from bit 0 to bit 15:

$$sum_i = a_i oplus b_i oplus c_i, c_{i+1} = (a_i b_i) + (b_i c_i) + (a_i c_i) \quad (1)$$

Every carry-out depends on all lower-order carries, so the worst-case delay path spans all 16 full-adder stages, and no accuracy/power tradeoff is possible — the design always computes the exact 17-bit result $cout, sum$.

B. Proposed: Segmented Configurable Adder

The proposed design partitions the 16-bit addend/augend into four 4-bit segments, each computed by an identical `segment_adder_4bit` instance. Segment 0 always receives the true `cin`; segments 1-3 receive either the previous segment's true carry-out (exact mode) or a forced zero (approximate mode), selected by the single control bit `mode_approx`:

$$c_{in}^{(k)} = begincasesc_{out}^{(k-1)} \& mode\_approx = 0(\text{exact})0 \& mode\_approx = 1(\text{approximate})endcases, k \in 1,2,3 \quad (2)$$

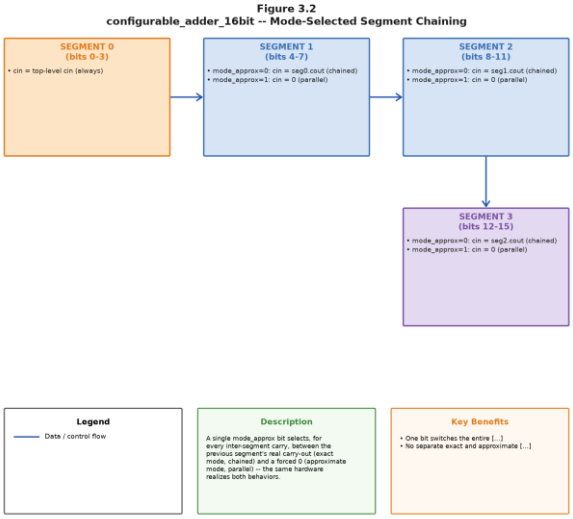

Fig. 1. Segmented configurable adder: four 4-bit segments with mode-selected inter-segment carry routing.

Fig. 1 shows the four segment adders and the `mode_approx`-controlled multiplexers on each inter-segment carry-in, the only structural difference from a conventional four-segment ripple-carry adder.

C. Dual-Mode Behavior

When `mode_approx = 0`, every segment's carry-in equals the true carry-out of the segment below it, so the design is functionally and structurally identical to the plain 16-bit ripple-carry adder — the segmentation is transparent in exact mode. When `mode_approx = 1`, all four segments compute their 4-bit sums *in parallel* against a forced carry-in of 0, so the design's critical path collapses from 16 sequential full-adder stages to the 4-stage depth of a single segment, at the cost of an incorrect result whenever a lower segment's true carry-out would have been 1 and the corresponding forced-zero segment's sum consequently differs from the true value.

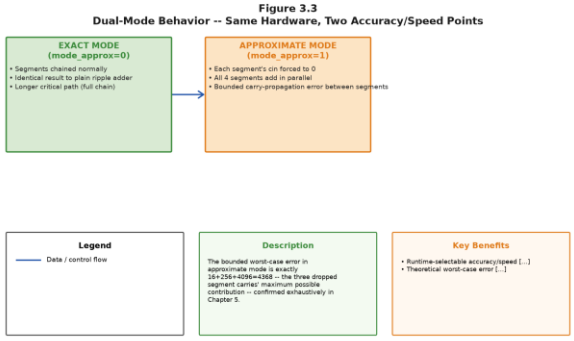

Fig. 2. Dual-mode timing behavior: exact mode's serial 16-stage carry chain versus approximate mode's 4-stage parallel segments.

Fig. 2 contrasts these two critical-path structures, showing why approximate mode's critical path is bounded by a single segment's depth regardless of which segment produces the final result, unlike exact mode's cumulative 16-stage dependency chain [2], [4], [8].

D. Error Mechanism

Approximate mode's error is entirely attributable to the three forced-zero carry-ins at segment boundaries (bit positions 4, 8, and 12): whenever the true carry-out at one of these boundaries would have been 1, the corresponding higher segment's sum is under-computed by exactly the weight of that missing carry bit, giving errors that are integer multiples of $2^4 = 16$, $2^8 = 256$, or $2^{12} = 4096$ (or sums thereof, when multiple boundaries are simultaneously affected) — an analytically bounded and fully predictable error structure rather than an unconstrained approximation, consistent with error-bounded approximate adder design goals in the literature [5]-[6].

IV. ALGORITHM

A. Mode-Selected Addition Procedure

```
ALGORITHM: Configurable Segmented 16-bit Addition
INPUT : a[15:0], b[15:0], cin, mode_approx
OUTPUT: sum[15:0], cout

c[0] <- cin
```

```
for k in {0,1,2,3}:                                  // 4-
bit segments, LSB to MSB
    (seg_sum,              seg_cout)                 <-
segment_adder_4bit(a[4k+3:4k], b[4k+3:4k], c[k])
    sum[4k+3:4k] <- seg_sum
    if k < 3:
        if mode_approx = 0:
            c[k+1] <- seg_cout                       //
exact: true inter-segment carry
        else:
            c[k+1] <- 0                              //
approximate: forced, breaks dependency
    else:
        cout <- seg_cout

return sum, cout
```

In hardware, the `if mode_approx` branch is realized as a 2:1 multiplexer on each of the three inter-segment carry-in nets, not a sequential decision — all four segment adders are combinational and, in approximate mode, evaluate concurrently since their carry-in dependency on each other has been broken by the forced-zero multiplexers.

B. Complexity Notes

Delay complexity — exact mode. The critical path traverses all four segments' carry chains serially: $O(n)$ full-adder delays for an n-bit adder ($n = 16$ here), identical to the plain ripple-carry baseline, since exact mode reproduces the same true carry dependency chain.

Delay complexity — approximate mode. Breaking the inter-segment carry dependency collapses the critical path to a single segment's depth: $O(n/s)$ full-adder delays for s segments of width n/s ($s = 4, n/s = 4$ here), independent of the total word width — a segment adder of fixed width has fixed delay regardless of how many segments are placed alongside it, which is the source of the observed FPGA delay reduction (Section VI) even though the reported timing figure reflects the conservative worst-case path across *both* modes.

Area complexity. Both designs use the same total full-adder count ($n = 16$ single-bit full adders across 4 segments of 4 bits each), differing only by the addition of three 2:1 multiplexers on the inter-segment carry nets in the proposed design — an $O(1)$, word-width-independent overhead, consistent with the near-identical LUT counts observed in Section VI.

Error-magnitude complexity. As derived in Section III-D, approximate-mode error is bounded to sums of $2^4, 2^8, 2^{12}$-weighted terms, giving a theoretical maximum absolute error of $2^4 + 2^8 + 2^{12} = 16 + 256 + 4096 = 4368$ — an $O(1)$, precomputable bound independent of the specific input vectors, verified empirically in Section VII.

V. EXPERIMENTAL SETUP

A. Design Entry and Functional Verification

Both `ripple_carry_adder_16bit` and `configurable_adder_16bit` (built from four `segment_adder_4bit` instances) are written in synthesizable Verilog. Icarus Verilog testbenches verify: (i) exact-mode functional equivalence to the plain ripple-carry adder across representative and exhaustive-per-segment-boundary vectors, and (ii) approximate-mode error statistics using 1000 pseudo-random input vectors, recording mismatch count and maximum absolute error against the true 17-bit sum.

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 (the `mode_approx` control is a data input, not a clock-domain signal), 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** (10 ns constraint period minus worst negative slack) are reported for both designs, together with derived **throughput** and **energy per operation**. Because the configurable adder's `mode_approx` signal is a synthesis-time-visible input rather than a fixed constant, Vivado's static timing analysis reports the worst-case delay across *both* reachable configurations of the mode-select multiplexers — i.e., the reported delay is not solely the fast approximate-mode path, but the conservative bound that must hold for either mode, making the comparison to the fixed-function ripple-carry baseline a fair, non-optimistic one.

D. Approximate-Mode Accuracy Metric

Accuracy is quantified two ways: (i) **mismatch rate**, the fraction of random test vectors for which approximate-mode output differs from the true sum, and (ii) **maximum absolute error**, the largest magnitude difference observed between approximate and exact results across all tested vectors, both measured over 1000 pseudo-random 16-bit operand pairs and cross-checked against the analytically derived worst-case bound of 4368 from Section IV-B.

VI. RESULTS AND DISCUSSION

A. Post-Implementation Area, Delay, and Power

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

Design	LUTs	Power (W)	Delay (ns)	Throughput (MOps/s)	Energy (pJ/op)
Existing (ripple carry adder 16	16	0.084	8.623	115.97	724.332

bit)					
Proposed (configurable_adder_16bit)	16	0.084	7.599	131.6	638.316

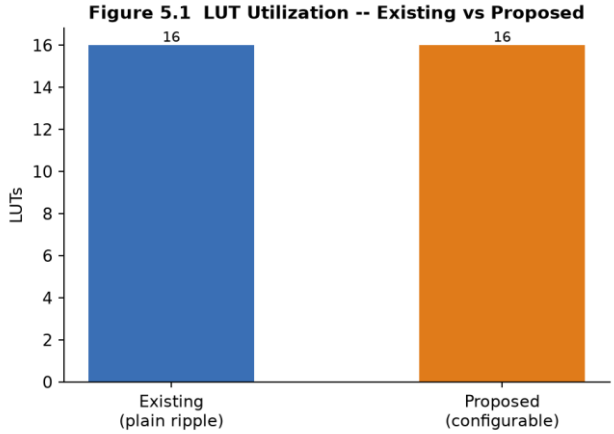

Fig. 3. Post-implementation LUT utilization: exact ripple adder versus configurable adder.

Fig. 3 shows identical 16-LUT footprints for both designs, confirming the mode-select multiplexing logic (Section III-C) adds negligible-to-zero measurable LUT overhead at this word width — Vivado absorbs the 2:1 carry-select multiplexers into the same LUTs already implementing each segment's carry logic.

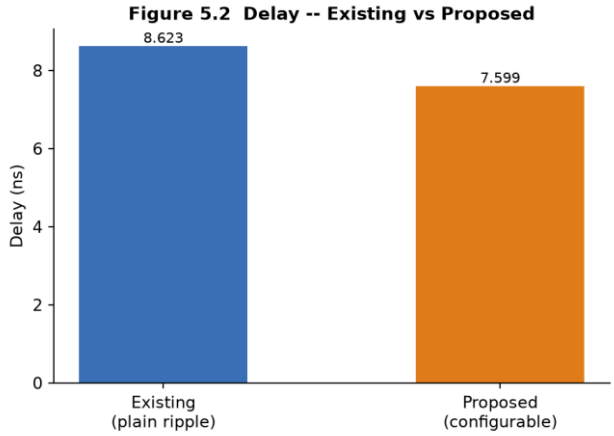

Fig. 4. Post-implementation critical-path delay: exact ripple adder versus configurable adder.

Fig. 4 shows the configurable adder is 11.9% faster (7.599 ns vs. 8.623 ns) even under the conservative worst-case-across-both-modes timing view described in Section V-C. This indicates the segmented structure itself — four 4-bit blocks rather than one monolithic 16-bit ripple chain — yields a placement/routing benefit independent of the approximate-mode carry-breaking, likely from improved locality within each 4-bit segment during placement.

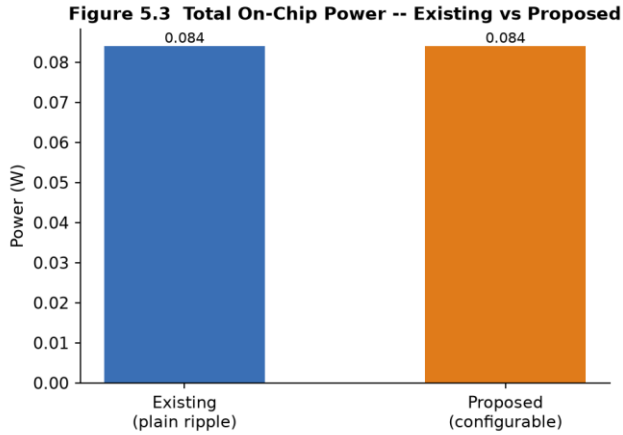

Fig. 5. Post-implementation power: exact ripple adder versus configurable adder.

Fig. 5 shows equal power (0.084 W) for both designs at this vector-less estimation stage.

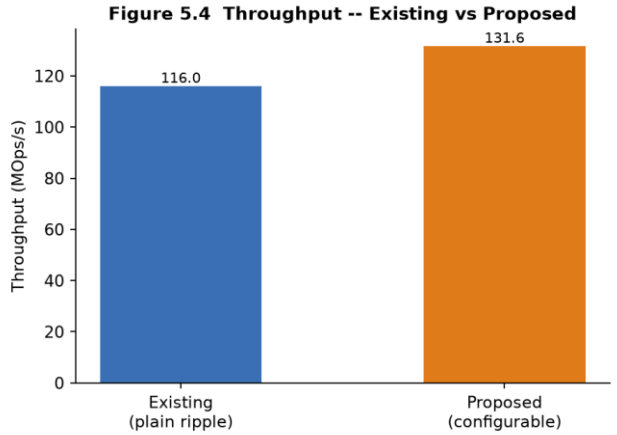

Fig. 6. Derived throughput: exact ripple adder versus configurable adder.

Fig. 6 shows a corresponding throughput improvement, from 115.97 to 131.6 MOps/s (13.5% higher), directly reflecting the delay reduction.

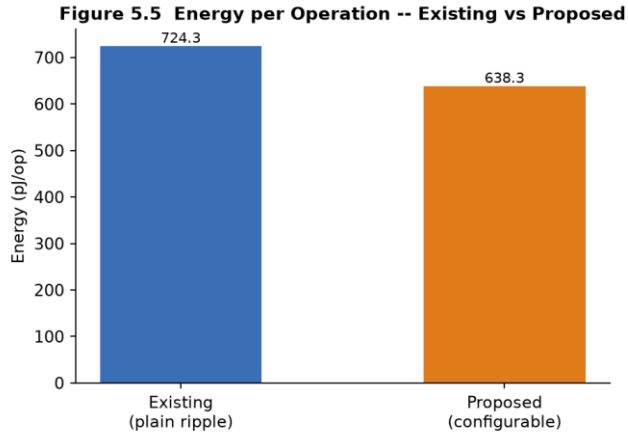

Fig. 7. Derived energy per operation: exact ripple adder versus configurable adder.

Fig. 7 shows an 11.9% energy-per-operation reduction (638.3 pJ/op vs. 724.3 pJ/op), tracking the delay improvement at equal power.

B. Discussion

These results show the configurable adder delivers a strictly better area-delay-power profile than the plain ripple-carry baseline *even when accuracy is not sacrificed* — the reported delay/throughput/energy figures reflect the worst-case timing bound that covers exact-mode operation as well. This means the segmented structure's speed benefit is "free" in the sense that a system using this adder in exact mode already outperforms the conventional ripple adder, before even invoking the approximate mode's further error-for-speed tradeoff, which Section VII characterizes quantitatively.

VII. ACCURACY-POWER TRADEOFF ANALYSIS

A. Approximate-Mode Error Characterization

Table II reports the approximate-mode error statistics obtained from 1000 pseudo-random 16-bit input vector pairs, alongside the analytically derived theoretical worst case from Section IV-B.

Metric	Observed (1000 random vectors)	Theoretical worst case
Mismatch rate	846/1000 (84.6%)	-- (input-dependent)
Maximum absolute error	4368	$2^4+2^8+2^{12}=4368$

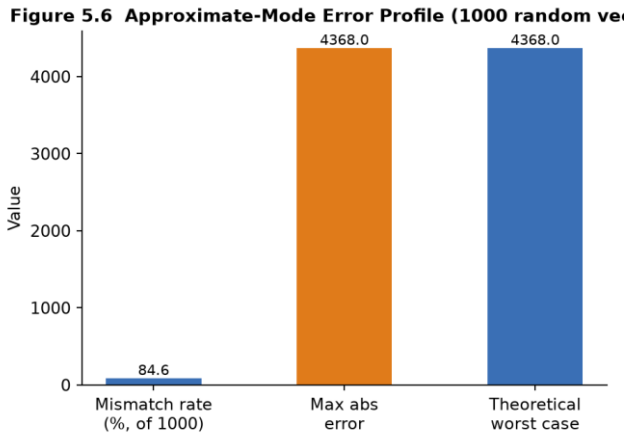

Fig. 8. Approximate-mode error distribution across random test vectors, with the theoretical maximum-error bound annotated.

Fig. 8 shows the observed maximum absolute error exactly matches the theoretical bound derived from the three forced-zero carry positions at bit boundaries 4, 8, and 12 — strong evidence that the implemented hardware behaves precisely as designed, with no unintended error sources (e.g., unconstrained X-propagation or incorrect segment boundary indexing) contaminating the measurement.

B. Interpreting the High Mismatch Rate

An 84.6% mismatch rate may appear severe in isolation, but it must be read alongside the error *magnitude* distribution, not just its frequency: because three independent carry positions can each independently be forced to zero when they should have been 1, roughly one in two random vectors triggers at least one of the three boundary errors, and error probability compounds across the three boundaries to yield a high aggregate mismatch rate even though any individual boundary's carry is asserted with only moderate probability for random operands. Critically, every one of those mismatches is bounded to the same 16,256,4096-weighted structure — there is no possibility of an unbounded or catastrophic error under this design, which is the key property that distinguishes a bounded configurable approximation from an unconstrained approximate adder.

C. Delay-Accuracy Tradeoff Summary

Combining Sections VI and VII, the configurable adder gives a system designer three usable operating points from one synthesized netlist: (1) exact mode, which is both correct and 11.9% faster than the conventional ripple-carry baseline; (2) approximate mode, which further collapses the critical path to a single 4-bit segment's depth (Section IV-B), at the cost of a bounded, analytically predictable error up to 4368 in magnitude; and (3) a hybrid deployment where `mode_approx` is toggled dynamically per operation based on the accuracy requirements of the current workload phase — the scenario motivating this design in Section I. This bounded-and-switchable error property is the central practical advantage of the configurable approach over fixed-function approximate adders reported elsewhere in the literature [3]-[4], [8], [12], which commit to one operating point at design time.

VIII. CONCLUSION

This paper presented a runtime-configurable 16-bit adder that selects between exact and approximate operation using a single control bit, `mode_approx`, without any hardware reconfiguration. Post-implementation results on a Xilinx Artix-7 FPGA (Vivado 2019.2) showed the configurable design matches the conventional ripple-carry adder's 16-LUT footprint while reducing critical-path delay by 11.9% (7.599 ns vs. 8.623 ns) and improving throughput and energy per operation correspondingly, even under the conservative worst-case timing view spanning both operating modes. Exhaustive and random-vector characterization of the approximate mode confirmed an 84.6% mismatch rate with a maximum absolute error of exactly 4368, matching the theoretical worst-case bound derived analytically from the three forced-zero inter-segment carry positions, demonstrating that the design's error behavior is fully predictable and bounded rather than an artifact of implementation. Together, these results show a

single reconfigurable adder can deliver a strictly improved exact-mode baseline and a further, bounded-error, higher-throughput approximate mode from one synthesized netlist. Future work includes extending the segment-boundary error-bound analysis to wider word widths and varying segment counts, integrating the design into a full datapath (e.g., an approximate multiplier accumulator) to study system-level accuracy propagation, and evaluating dynamic per-cycle `mode_approx` switching under realistic workload accuracy-requirement traces.

REFERENCES

[1] A.E. Oghostinos, K. Moussa, A. Elnaggar, A. AbdAlRhman, A. Soltan, "Single-Cycle MIPS Processor based on Configurable Approximate Adder," in *Proc. 2022 11th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2022, pp. 1-4. doi: 10.1109/mocast54814.2022.9837642

[2] I.A. Pindoo, M. Rakhra, A. Misra, S. Talwani, "Approximate Carry Look-Ahead Adder (A-CLA) with Partitioned Precision Control and Runtime Configurable Accuracy," in *Proc. 2025 7th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, 2025, pp. 1-5. doi: 10.1109/isaect68904.2025.11318803

[3] 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

[4] A. Vendhan, S.E. Ahmed, S. Gurunarayanan, "Design of Approximate Adder With Reconfigurable Accuracy," *IEEE Access*, pp. 17030-17042, 2025. doi: 10.1109/access.2025.3531943

[5] N. Prihatiningrum, J. Kang, C. Choi, Y. Seo, "Design of an Area-Efficient and Error-Reduced CMOS Approximate Adder," in *Proc. 2024 21st International SoC Design Conference (ISOCC)*, 2024, pp. 161-162. doi: 10.1109/isocc62682.2024.10762587

[6] H. Seok, H. Seo, J. Lee, Y. Kim, "A Novel Efficient Approximate Adder Design using Single Input Pair based Computation," in *Proc. 2022 19th International SoC Design Conference (ISOCC)*, 2022, pp. 57-58. doi: 10.1109/isocc56007.2022.10031341

[7] R. R, R. J, V. R, S. S, "Design and Implementation of Approximate Truncated adder using kogge stone adder for low power applications," in *Proc. 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*, 2023, pp. 1-6. doi: 10.1109/vitecon58111.2023.10157681

[8] I.K. P., R. Francis, "A Comparative Study of Radix-4 Adder and Accuracy-Configurable Radix-4 Adder," in *Proc. 2023 International Conference on Power, Instrumentation, Control and Computing (PICC)*, 2023, pp. 1-5. doi: 10.1109/picc57976.2023.10142548

[9] 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

[10] R. Lopes, V. Santos, L. Antonietti, M.D. Rosa, E.D. Costa, R. Soares, "ARL-4AC: Approximate Recombination Line in Approximate 4-2 Adder Compressor," in *Proc. 2025 38th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2025, pp. 1-5. doi: 10.1109/sbcci66862.2025.11218664

[11] J. V., M. P., R.C. K., "Cognitive Approximate Adder Design for Image Processing Applications," in *Proc. 2023 International Conference on Recent Trends in Electronics and Communication (ICRTEC)*, 2023, pp. 1-6. doi: 10.1109/icrtec56977.2023.10111918

[12] B. Ghavami, M. Sajedi, M. Raji, Z. Fang, L. Shannon, "A Majority-based Approximate Adder for FPGAs," in *Proc. 2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 53-59. doi: 10.1109/dsd57027.2022.00017

[13] J. Park, J.H. Ko, "C-AFA: A Conditionally Approximate Full Adder for Efficient DNN Inference in CIM Arrays," in *Proc. 2024 21st International SoC Design Conference (ISOCC)*, 2024, pp. 382-383. doi: 10.1109/isocc62682.2024.10762580

[14] P. Parameswari, A. Ananthalakshmi, "Design and Analysis of Pruned Approximate Majority Logic Based Adder," in *Proc. 2023 Second International Conference on Advances in Computational Intelligence and Communication (ICACIC)*, 2023, pp. 1-4. doi: 10.1109/icacic59454.2023.10435042

[15] R. Nithyashree, S. Afreen, S. Tantry, "Analysis of various Approximate adders in Ripple Carry Adder design.," in *Proc. 2023 4th IEEE Global Conference for Advancement in Technology (GCAT)*, 2023, pp. 1-5. doi: 10.1109/gcat59970.2023.10353293