An Energy-Efficient Approximate Posit Multiply–Divide Unit

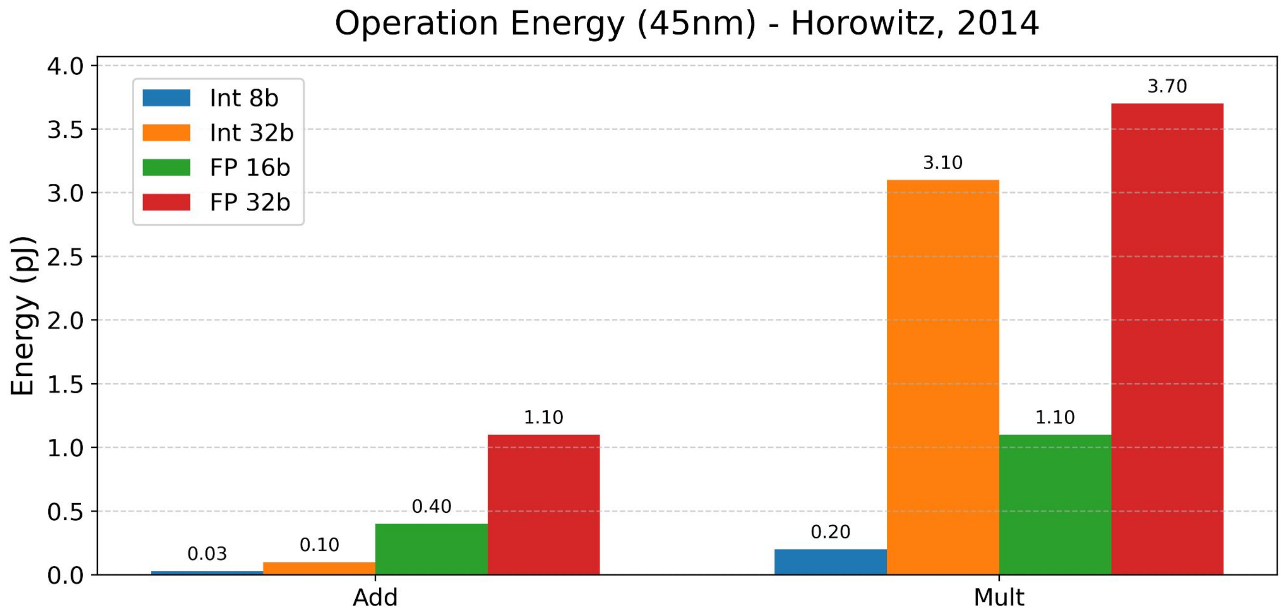
Rishi Thotli
Aditya Anirudh Jonnalagadda
Rushabh Hulsurkar
Uppugunduru Anil Kumar
Sreehari Veeramachaneni
Syed Ershad Ahmed
John L. Gustafson

Abstract

- Division = slowest arithmetic operation; consumes more power & area.
- Traditional fast dividers → iterative reciprocal + multiply.
- We propose an **approximate division** using a **LUT-based reciprocal** integrated into the **posit decoder**.
- Unified unit handles exact multiplication and approximate division.
- Achieves ~77% PDP reduction, ~46% area reduction vs existing designs when performing inexact division for 16b Posits with a roughly ~0.27% MED from exact results.

Background: Multiplication and Division

- Multiplication → core operation in DSP, ML, graphics.
- Division → less frequent but latency-bound.
- Both dominate energy footprint of arithmetic pipelines.



Division in Hardware (Floating-Point Systems)

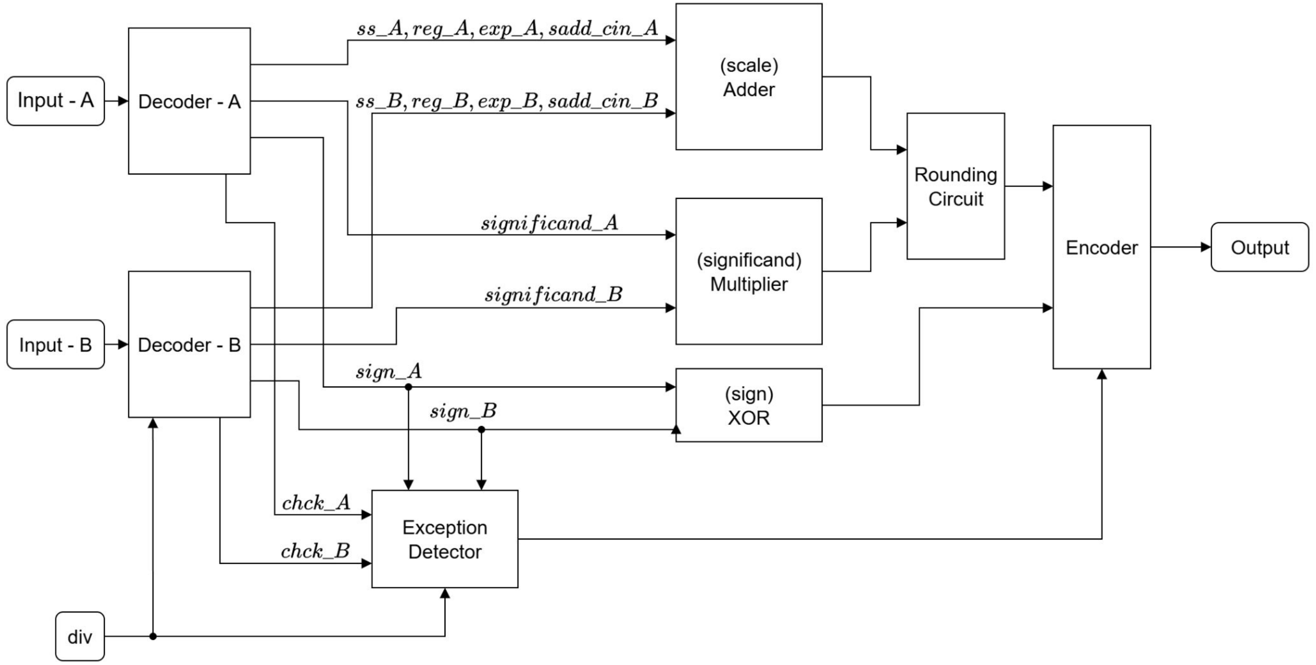
- Realized via:
  - **Digit recurrence** (bit-by-bit quotient).
  - **Functional iteration** (e.g., Newton–Raphson).
- Floating-point and Posit both use similar principles.

Goal of This Work

- Unified **posit multiply–divide unit** that:
  - Performs **exact multiplication** and **approximate division**.
  - Reduces **delay**, **power**, and **area** compared to existing designs.
  - Enables future exact division using iterative methods (e.g., NR).
- Serves as a **foundation** for either low-error workloads or iterative refinement.

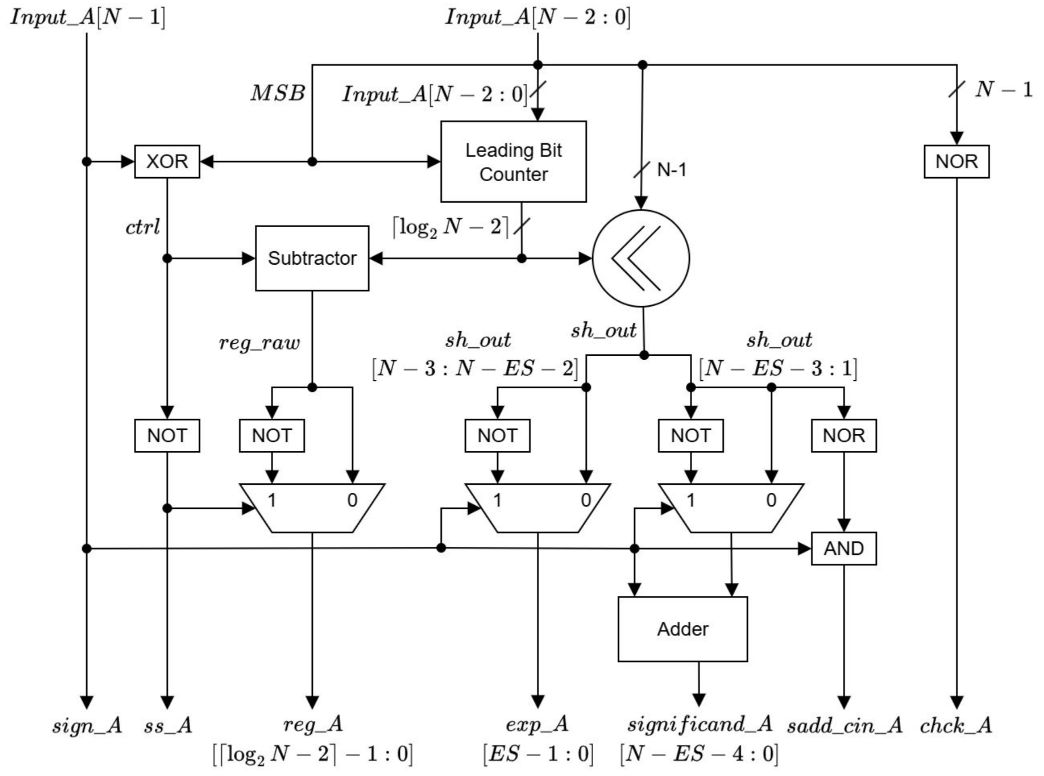
Proposed Architecture Overview

- Unified data path handles both operations.
- Reciprocal computation integrated in Decoder-B.



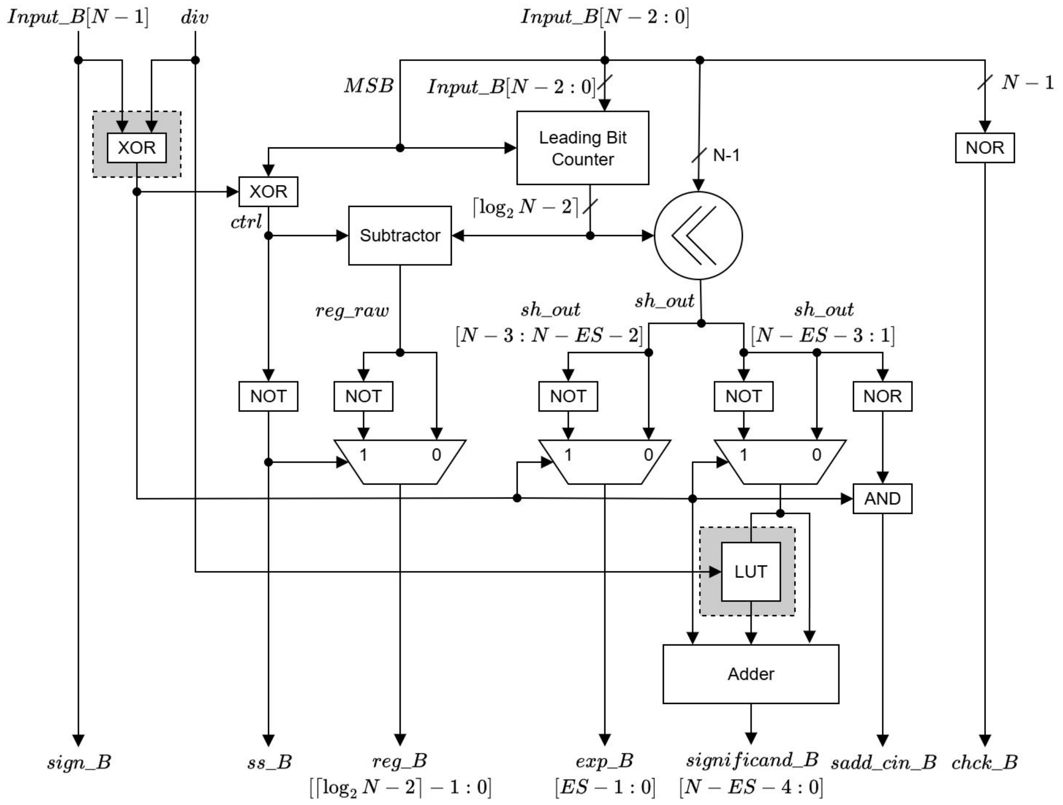
Decoder-A: Baseline

- Extracts sign, regime, exponent, fraction.
- ‘Parallel’ processing of fields wherever possible to minimize delay.
- Uses an internal control signal ctrl for keeping track of regime polarity.
- Exception detection integrated.

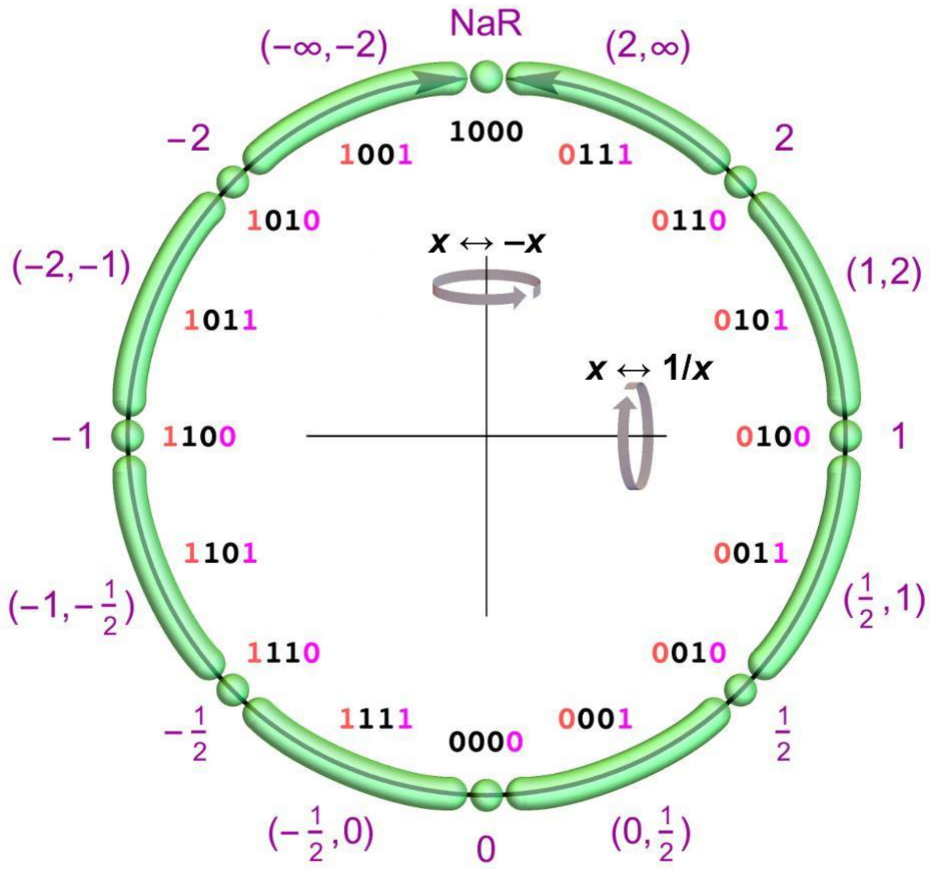


Decoder-B: Key Innovation

- Extends Decoder-A with reciprocal computation mode.
- Controlled by single bit div.
- Introduces error-correction LUT.
- Performs reciprocal estimation directly in the decoder.
- Allows for a partial overlap of reciprocal estimation and significand extraction, cutting down on total delay.

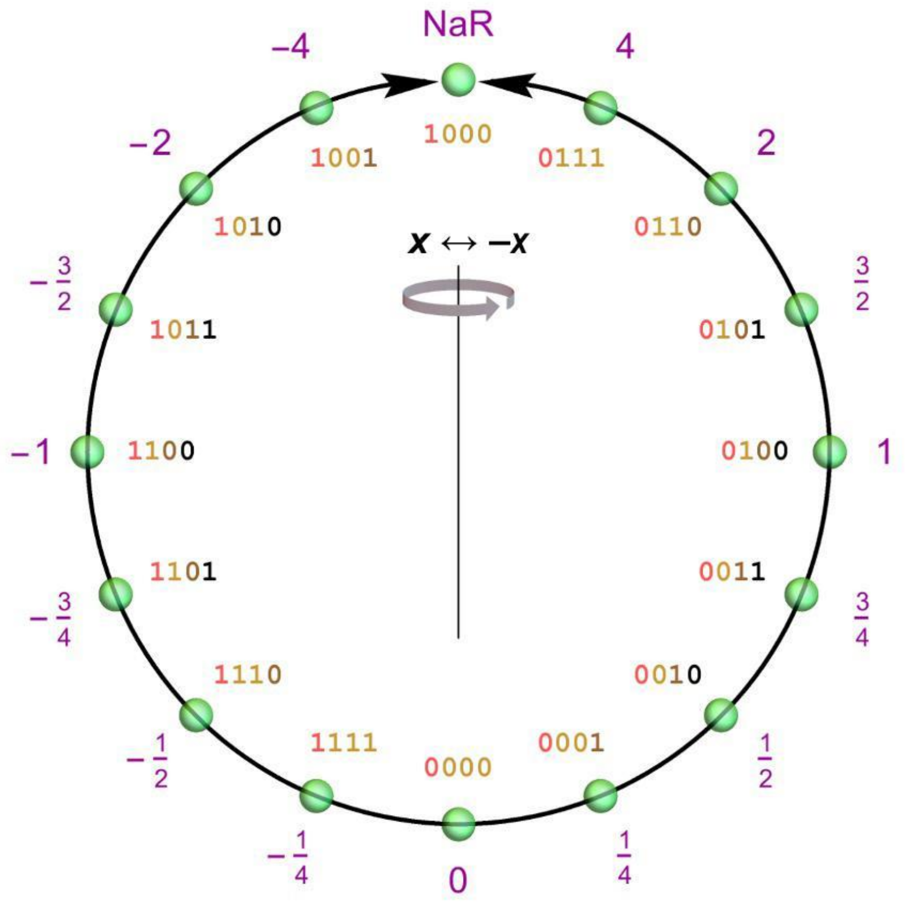


Reciprocal Computation – Core Idea



- Type II unums make reciprocation as perfect as negation
- This property also holds for *logarithmic* posits, where each binade is 2^{i+f} instead of $2^{i}\times(1+^{f})$
- Plus-minus-times-divide hardware needs only an add/subtract unit and a multiply/divide unit!

Reciprocal Computation – Core Idea

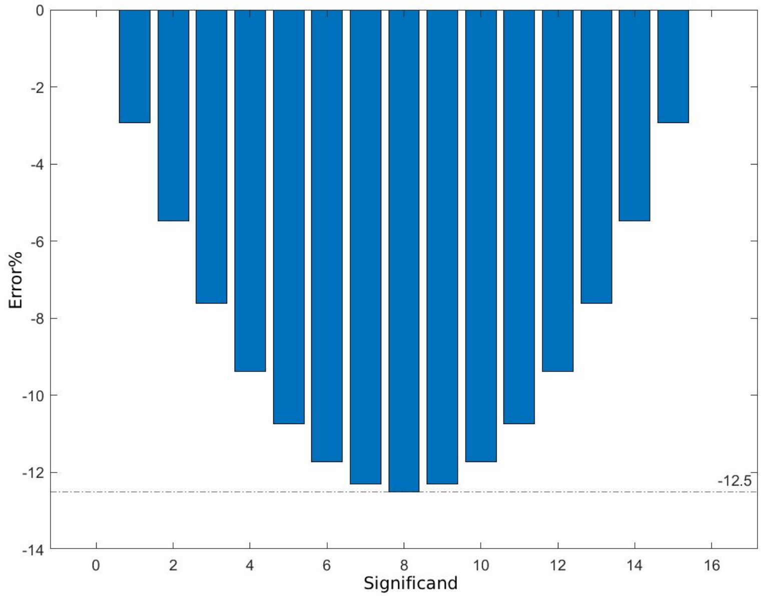


- When we use linear binades, we lose perfect reciprocals
- Now reciprocals are only perfect for powers of 2.
- Still, the other values are very close to being perfect…

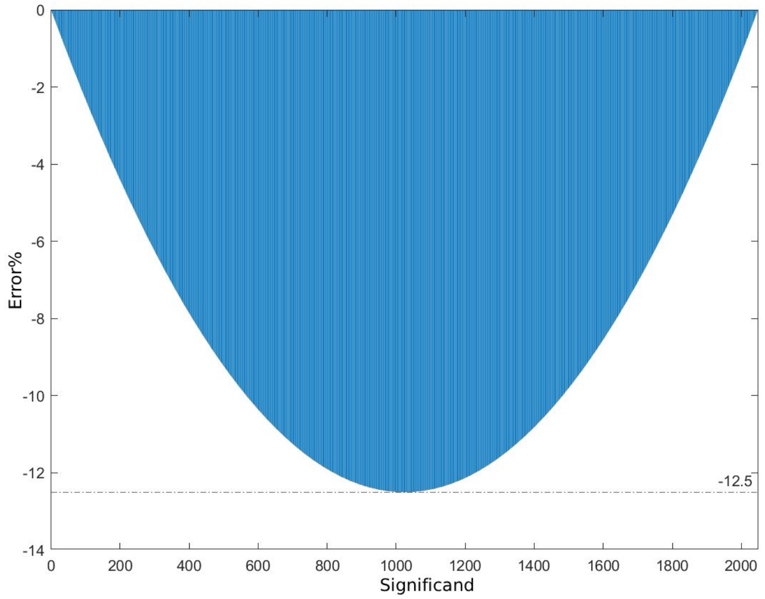
Can we apply a fast correction and again combine the multiply and divide units?

Reciprocal Computation – Core Idea

- Reciprocal approximated via **2’s complement** of input (sign held constant).
- Exact when fraction = 0.
- Linearizes 1/(1+f) ≈ 1–f/2 → small deviation corrected by LUT.



Error % plotted across all possible significands (4b)



Error % plotted across all possible significands (11b)

Error Correction Algorithm

- **Corrective term:**

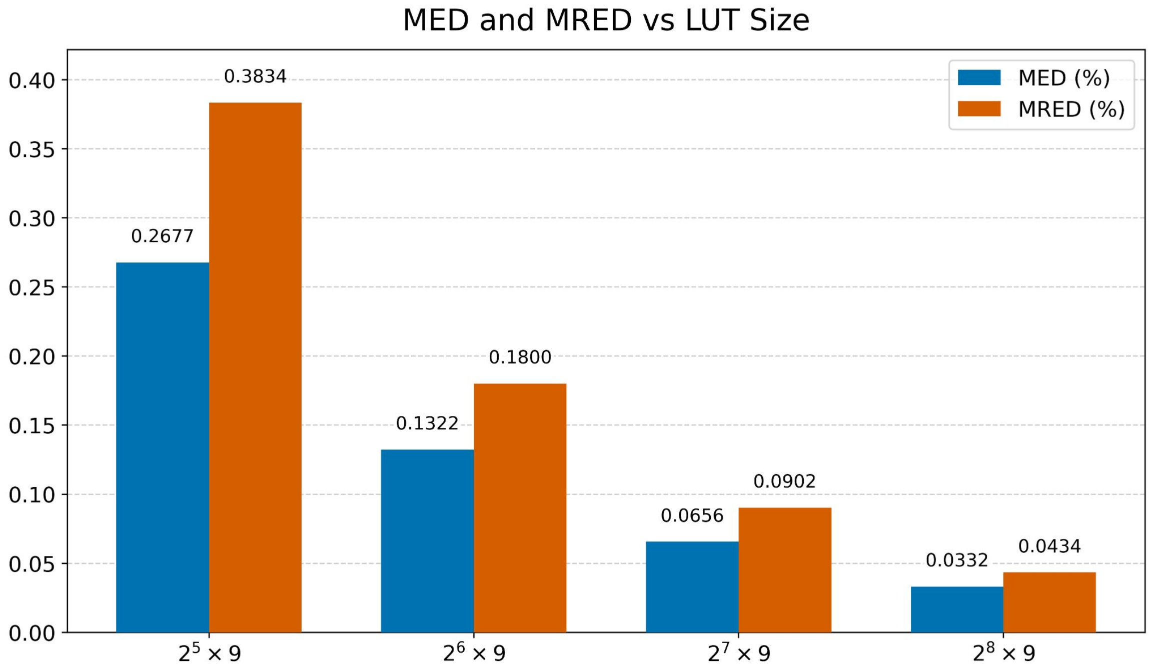
$$EC = \frac{f(1-f)}{(1+f)} 2^{\mathrm{FB}}$$

- **Steps:**
  - 2’s complement input.
  - Use MS fraction bits → LUT address.
  - Subtract EC value from processed fraction.
- Integer-rounded EC stored in LUT

LUT Accuracy & Tradeoffs

- Doubling LUT halves error (MED ↓, MRED ↓).
- $2^5 \times 9$: 0.38% MRED (baseline used for hardware result)

LUT Size	MED (%
2^5 x 9	0.2677
2^6 x 9	0.1322
2^7 x 9	0.0656
2^8 x 9	0.0332

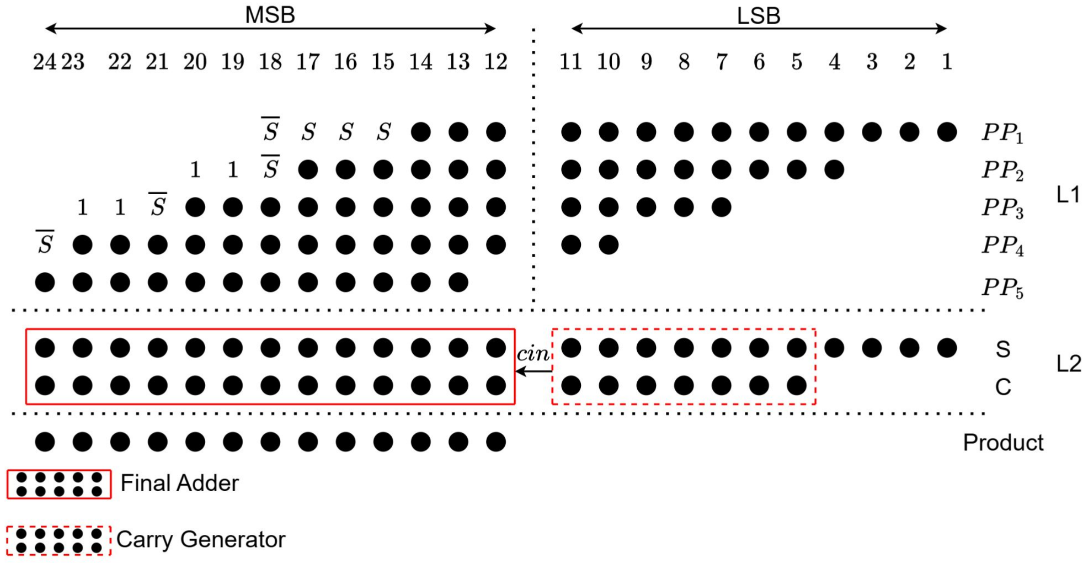


Comparison with Naive NR Seed

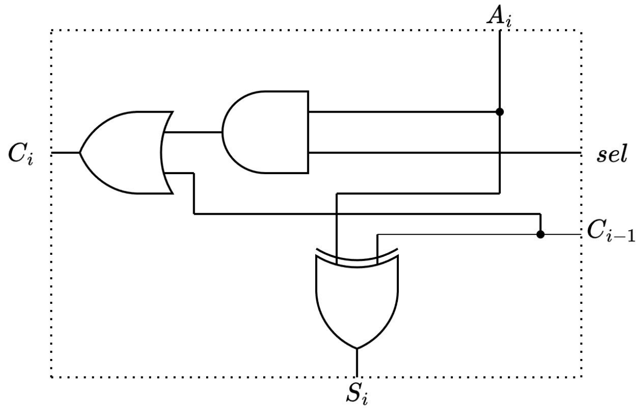
- NR-based designs (PACoGen) use LUT seed + iterations.
- Proposed method: 14× better seed accuracy when using the same LUT size.
- Potential to reduce NR iterations needed for exact division with some additional tricks.

Significand Multiplier

- Modified radix-8 Booth multiplier with custom power-gating.
- Streamlined 2’s complement logic → eliminates control circuitry.
- Reduces dynamic power and area.



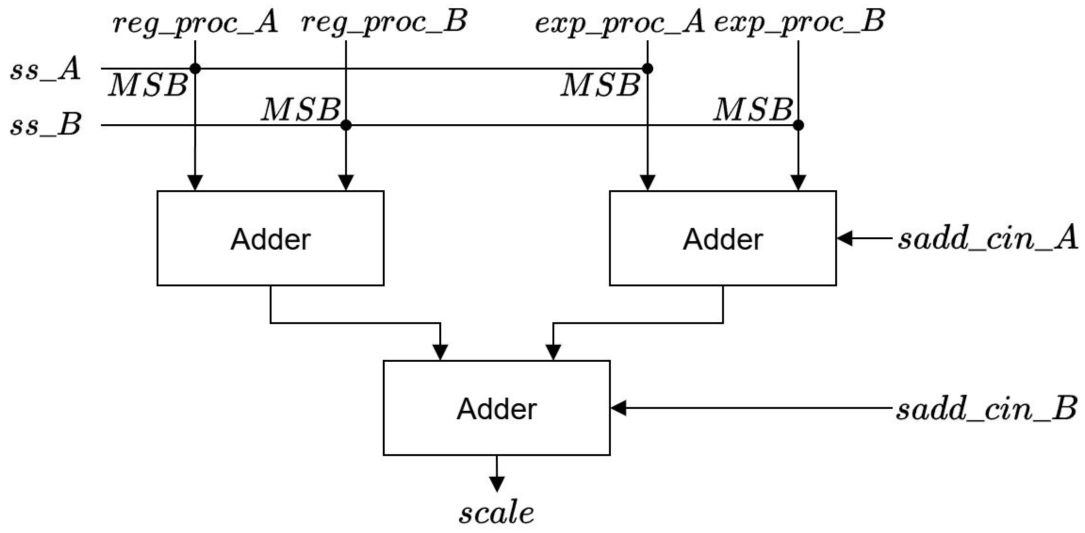
Partial product array of proposed significand multiplier



2's complement block used in PPG

Scale Adder

- Combines regime & exponent results from decoders.
- Creates a new intermediary format called ‘scale’.
- Performs addition in parallel to the Significand Multiplication and doesn’t add any extra delay in the critical path.



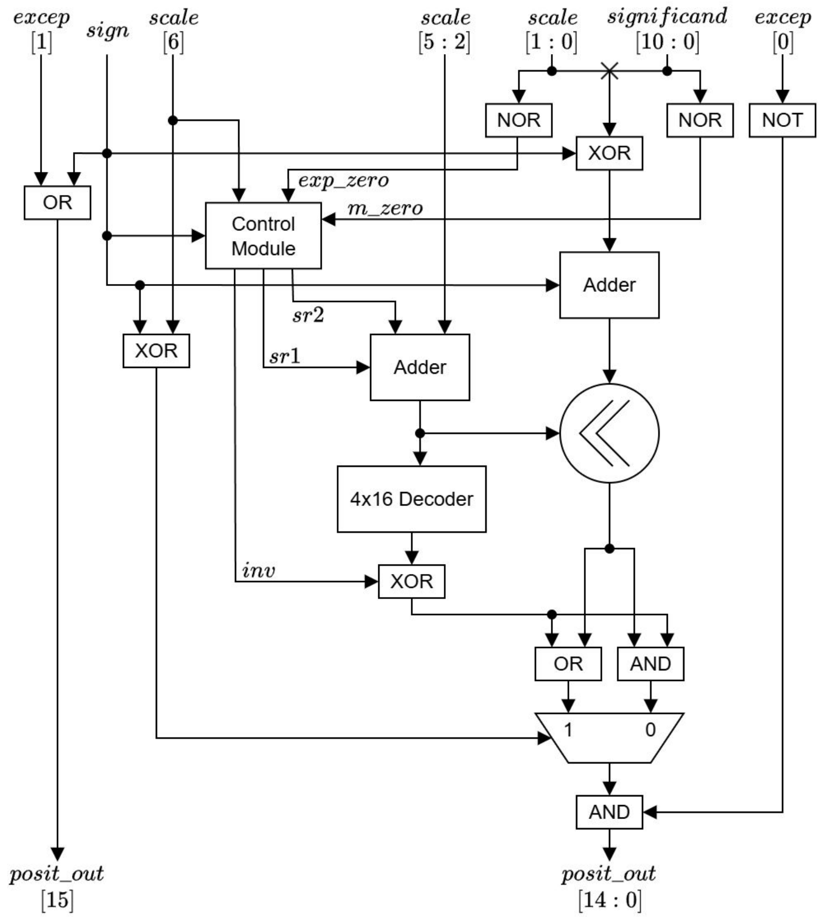
Exception Detector

- Handles NaR, zero, and invalid cases.
- Outputs 2-bit excep signal → Encoder uses for correction.

div	***sign_A***	***chck_A***	***sign_B***	***chck_B***	***Output***
x	x	0	x	0	Normal
0	0	1	x	0	Zero
0	x	0	0	1	Zero
0	x	0	1	1	NaR
1	x	0	1	1	Zero
1	0	1	0	1	NaR

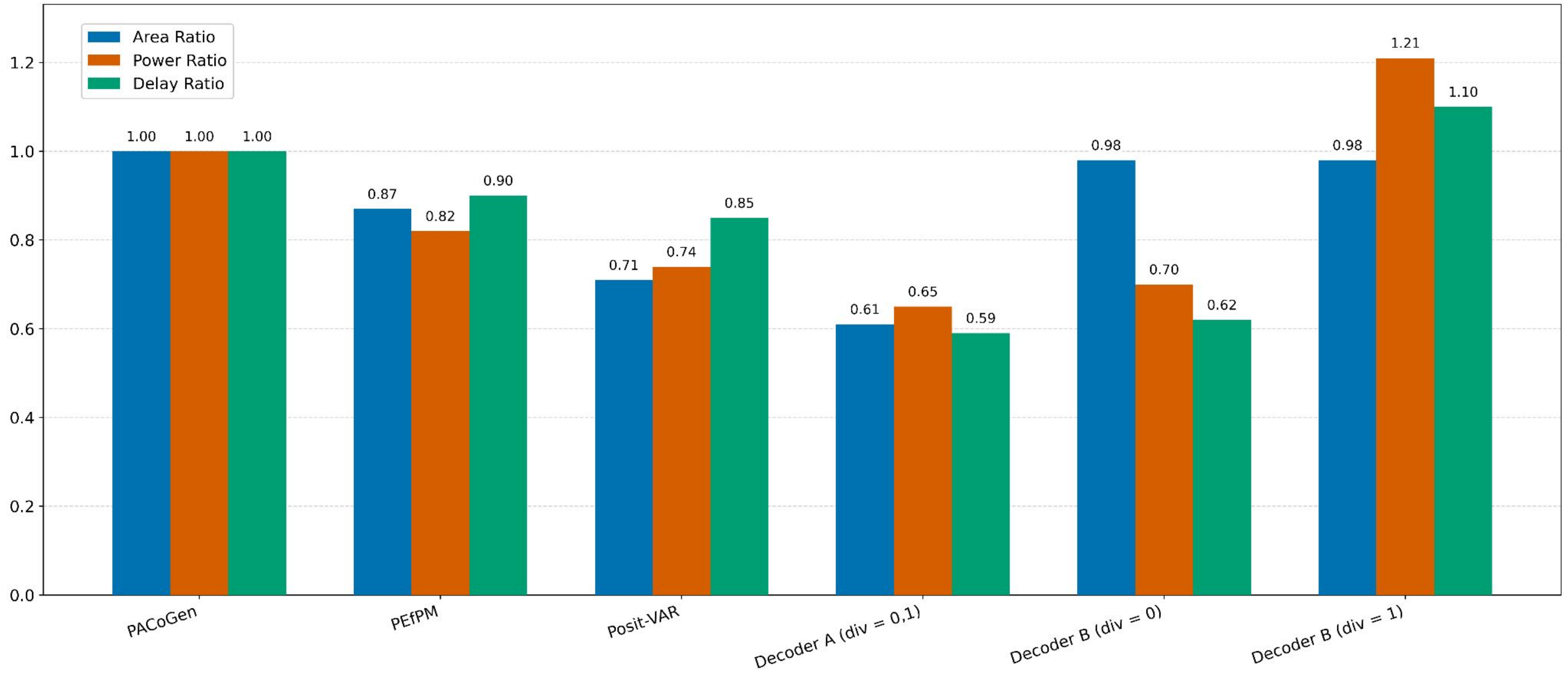
Encoder

- Merges sign, scale, and fraction into final posit.
- Heavy parallelization to minimize delay.
- Integrated exception correction.



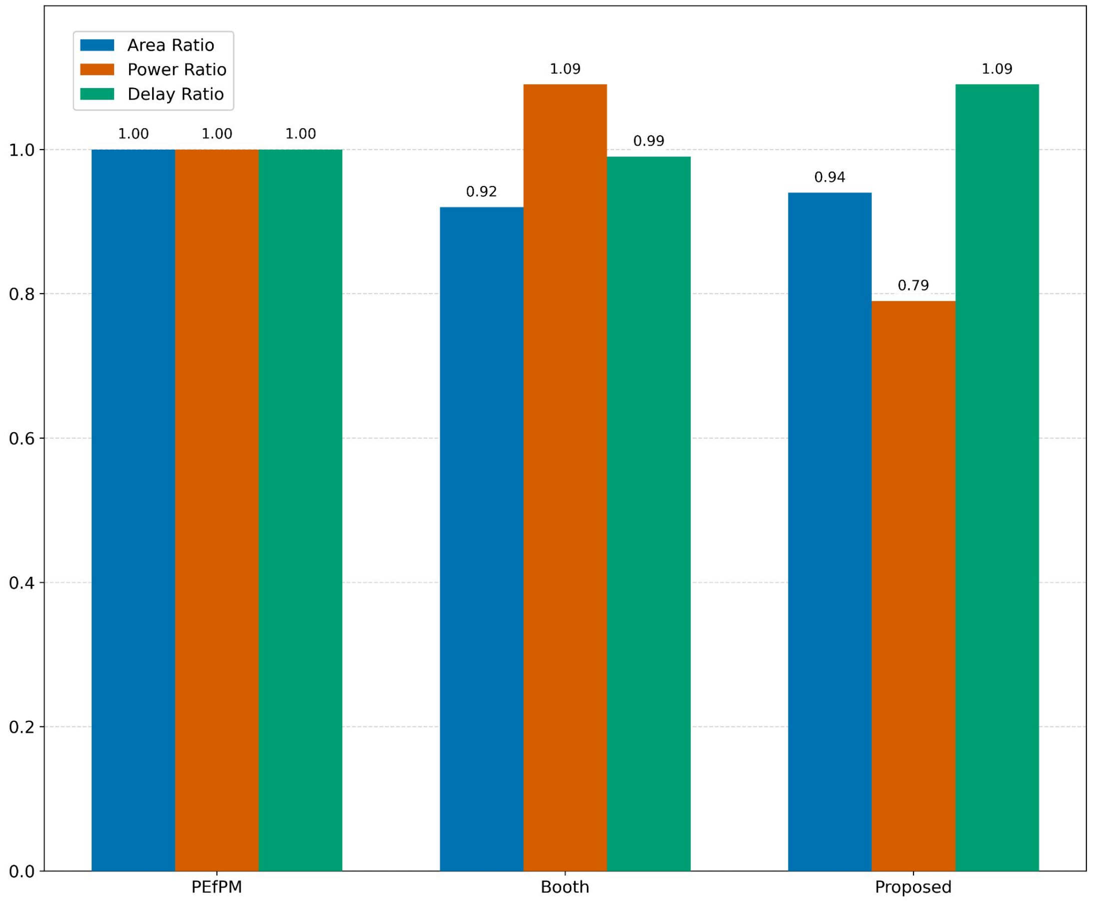
Decoder Performance

- Decoder-A: 31.5% lower PDP than next-best.
- Decoder-B: minimal overhead in multiply mode.



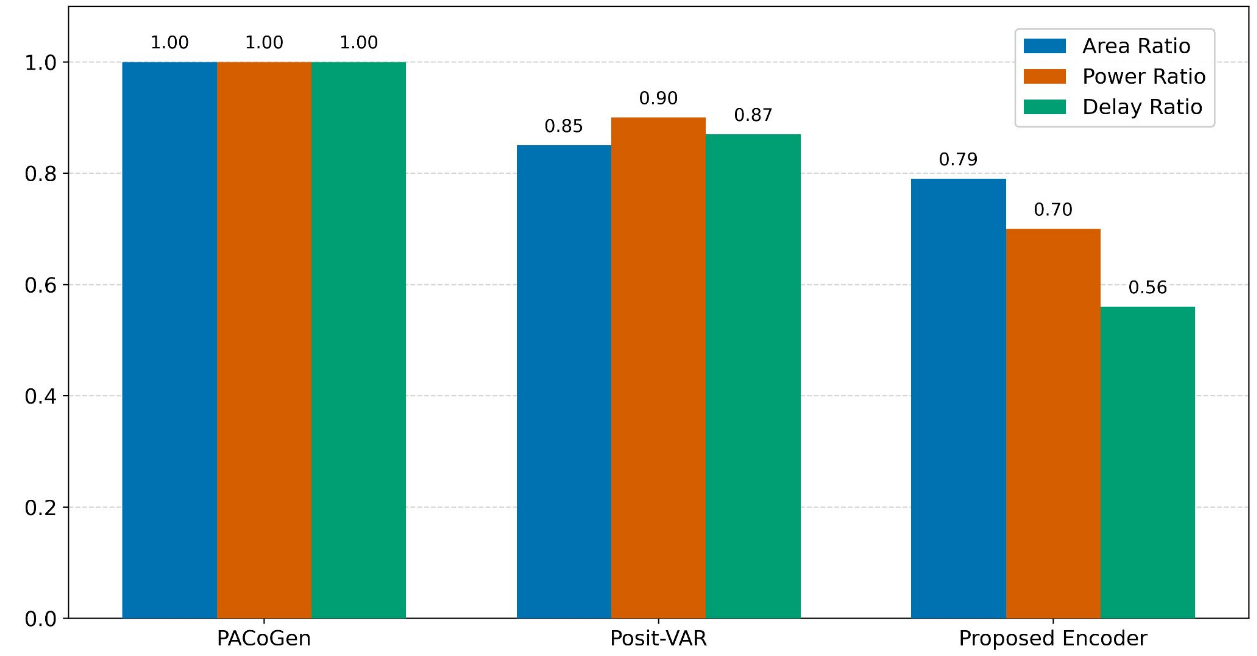
Multiplier Performance

- Significantly lower power
- Comparable area/delay



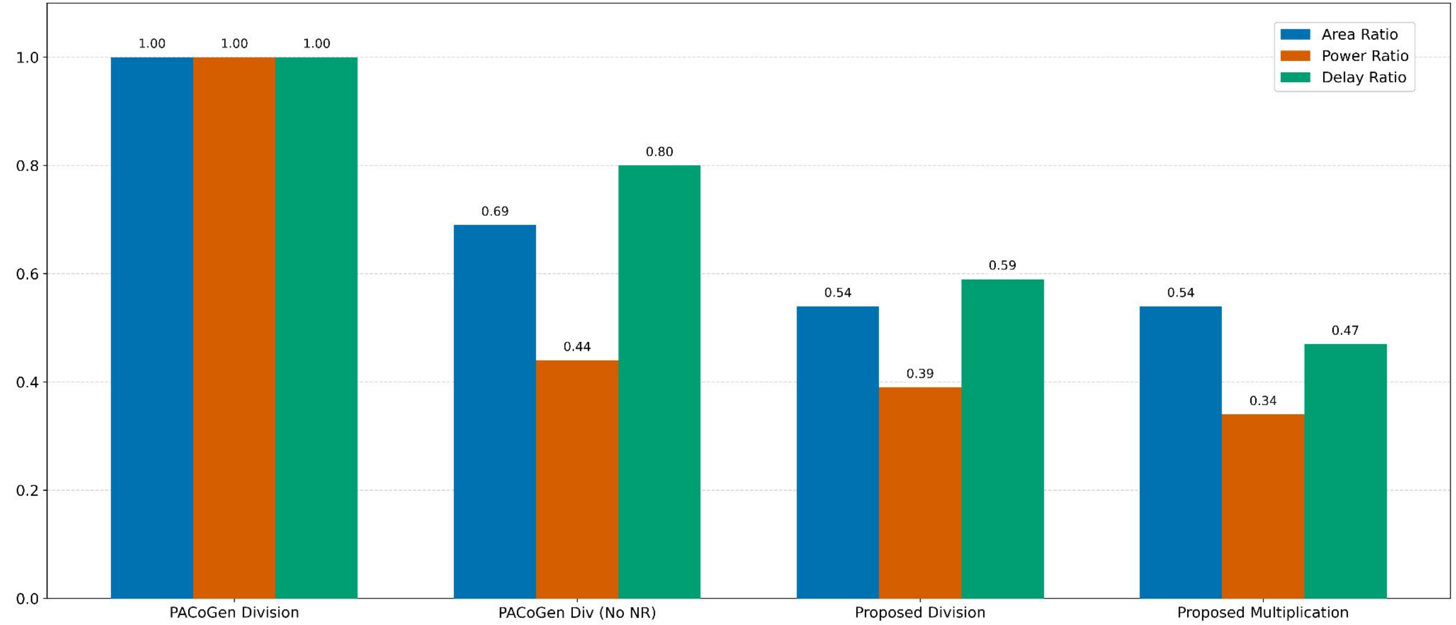
Encoder Performance

- Significant reduction in all three metrics compared to prior works.
- Delay ↓35%, Power ↓23%, Area ↓8% vs the next best (Posit-VAR)



Full Datapath Comparison

- Unified unit during 16b approximate division:
  - PDP ↓77%
  - Area ↓46%



Summary

- Unified posit multiply–divide unit.
- Integrated reciprocal logic → efficient starting/ending point for division.
- Major power and delay savings.
- Fully modular architecture (each block optimized).
- LUT can be customized to suit application error tolerance.
- Can be easily converted to perform exact division through conventional, iterative means.

Future Work

- Smarter LUT allocation, with denser EC values near midrange where the error is the greatest.
- Linear EC approximation near extremes to further free up LUT space for the middle.
- Combine planned enhancements and implement the scheme for constrained formats like B-Posits to further increase area budget for a larger LUT.
- Use improved seed accuracy to reduce NR iterations required for full-precision division.