Correctly Rounded Functions: From Pen-And-Paper to Formal Proofs

Paul Zimmermann (INRIA Nancy) and Wonyeol Lee (POSTECH)

FPTalks Annual Workshop, August 6, 2026

Evaluation Assurance Level

Common Criteria EAL levels		
	EAL7	Formally verified designed & tested
	EAL6	Semi formally verified designed & tested
	EAL5	Semi formally designed & tested
	EAL4	Methodically designed, tested & reviewed
	EAL3	Methodically tested & checked
	EAL2	Structurally tested
	EAL1	Functionally tested

Where Are We for Math Libraries?

EAL0: **No specification** on math libraries

IEEE 754-$\leq$2019

Where Are We for Math Libraries?

EAL3: **Correct rounding** (heavy testing) — RLIBM, LLVM libc, CORE-MATH, ... IEEE 754-2029?

EAL0: **No specification** on math libraries — IEEE 754-$\leq$2019

Where Are We for Math Libraries?

EAL5: Correct rounding (**pen-and-paper proofs**)	CORE-MATH-PROOF (this talk)
EAL3: **Correct rounding** (heavy testing)	RLIBM, LLVM libc, CORE-MATH, ... IEEE 754-2029?
EAL0: **No specification** on math libraries	IEEE 754-$\leq$2019

Where Are We for Math Libraries?

EAL7: Correct rounding (**formally-checked proofs**)	IEEE 754-2099?
EAL5: Correct rounding (**pen-and-paper proofs**)	CORE-MATH-PROOF (this talk)
EAL3: **Correct rounding** (heavy testing)	RLIBM, LLVM libc, CORE-MATH, ... IEEE 754-2029?
EAL0: **No specification** on math libraries	IEEE 754-$\leq$2019

Why Should We Care?

Goal of **correctly rounded implementations**.

For all $x \in \mathbb{F}$, $\texttt{f}(x) = \mathrm{rnd}(f(x))$. ($\texttt{f} : \mathbb{F} \to \mathbb{F}$ is impl'n, $f : \mathbb{R} \rightharpoonup \mathbb{R}$ is target.)

[1] Correctly Rounded Functions For Vector Applications: A Performance Study.
Anderson, Cornea, Stepin, Panu, ArXiv 2026. (`https://arxiv.org/pdf/2605.15547`)

[2] Correctly Rounded Vector Implementation of the Exponential Function in binary64 Arithmetic.
Brisebarre, Hubrecht, Lauter, Muller, Ruiz-Rohena, ARITH 2026.

Why Should We Care?

Goal of **correctly rounded implementations**.

For all $x \in \mathbb{F}$, $\texttt{f}(x) = \mathrm{rnd}(f(x))$. ($\texttt{f} : \mathbb{F} \to \mathbb{F}$ is impl'n, $f : \mathbb{R} \rightharpoonup \mathbb{R}$ is target.)

They are **more and more popular**.

Scalar: RLIBM, LLVM libc, CORE-MATH, GNU libc (10 binary64 functions), ...

Vectorized: Intel's upcoming library [1], ARITH 2026 paper on binary64 exp [2], ...

[1] Correctly Rounded Functions For Vector Applications: A Performance Study.
Anderson, Cornea, Stepin, Panu, ArXiv 2026. (`https://arxiv.org/pdf/2605.15547`)

[2] Correctly Rounded Vector Implementation of the Exponential Function in binary64 Arithmetic.
Brisebarre, Hubrecht, Lauter, Muller, Ruiz-Rohena, ARITH 2026.

Why Should We Care?

Goal of **correctly rounded implementations**.

For all $x \in \mathbb{F}$, $\mathtt{f}(x) = \mathrm{rnd}(f(x))$. ($\mathtt{f} : \mathbb{F} \to \mathbb{F}$ is impl'n, $f : \mathbb{R} \rightharpoonup \mathbb{R}$ is target.)

They are **more and more popular**.

Scalar: RLIBM, LLVM libc, CORE-MATH, GNU libc (10 binary64 functions), ...

Vectorized: Intel's upcoming library [1], ARITH 2026 paper on binary64 exp [2], ...

They often have **subtle correctness issues**.

$$\mathtt{sinh}(-5.014\ldots) = -1.2\mathtt{d45dac997dc5}_{(16)} \times 2^6$$
$$\mathrm{rnd}(\sinh(-5.014\ldots)) = -1.2\mathtt{d45dac997dc4}_{(16)} \times 2^6, \text{ and many more.}$$

[1] Correctly Rounded Functions For Vector Applications: A Performance Study.
Anderson, Cornea, Stepin, Panu, ArXiv 2026. (https://arxiv.org/pdf/2605.15547)

[2] Correctly Rounded Vector Implementation of the Exponential Function in binary64 Arithmetic.
Brisebarre, Hubrecht, Lauter, Muller, Ruiz-Rohena, ARITH 2026.

Related Work

- Floating Point Verification in HOL Light: The Exponential Function.
  Harrison, FMSD 2000.
- Formal Verification of Floating Point Trigonometric Functions.
  Harrison, FMCAD 2000.
- Towards a Correctly-Rounded and Fast Power Function in binary64 Arithmetic (Extended Version).
  Hubrecht, Jeannerod, Zimmermann, Rideau, Théry, ARITH 2024.
- On Automatically Proving the Correctness of math.h Implementations.
  Lee, Sharma, Aiken, POPL 2018.
- ... and many more.

The CORE-MATH-PROOF Project

Overview.

Goal: **Prove correctness** of CORE-MATH binary64 functions, by pen-and-paper.

By **Paul Zimmermann**, Olivia Chagot, Guillaume Melquiond, Cyprien Peignier.

Started just this year (2026).

The CORE-MATH-PROOF Project

Overview.

Goal: **Prove correctness** of CORE-MATH binary64 functions, by pen-and-paper.

By **Paul Zimmermann**, Olivia Chagot, Guillaume Melquiond, Cyprien Peignier.

Started just this year (2026).

Status.

Proofs done: `acos`, `acosh`, `asin`, `atanh` (ARITH 2026), `cbrt`, `cosh`, `sinh`.

Proofs in progress: `asinh`, `exp`, `expm1`, `tanh`.

Acknowledgments.

Sylvain Chevillard, Sungjin Hwang, Vincenzo Innocente, Claude-Pierre Jeannerod, Hyunsoo Lee, Wonyeol Lee, Pavel Panchekha, Robert Strandh, Adhemerval Zanella.

Proof Overview

Example: Proof of binary64 `atanh` (ARITH 2026).

Implementation (simplified).

$x \in \mathbb{F} \cap [0, 2^{-26}) \implies \cdots$

$x \in \mathbb{F} \cap [2^{-26}, 2^{-2}) \implies$ Compute $p(x) = x + c_3x^3 + \cdots + c_{21}x^{21}$ **using floats**.
Let $\mathtt{p}(x) \approx p(x)$ be the computed value. Return $\mathtt{p}(x)$.

$x \in \mathbb{F} \cap [2^{-2}, \infty) \implies \cdots$

Proof Overview

Example: Proof of binary64 `atanh` (ARITH 2026).

Implementation (simplified).

$x \in \mathbb{F} \cap [0, 2^{-26}) \implies \cdots$

$x \in \mathbb{F} \cap [2^{-26}, 2^{-2}) \implies$ Compute $p(x) = x + c_3x^3 + \cdots + c_{21}x^{21}$ **using floats**.
Let $\mathtt{p}(x) \approx p(x)$ be the computed value. Return $\mathtt{p}(x)$.

$x \in \mathbb{F} \cap [2^{-2}, \infty) \implies \cdots$

Claim: $\mathtt{atanh}(x) = \mathrm{rnd}(\mathrm{atanh}(x))$ for all $x \in \mathbb{F}$. (Assume round-to-nearest mode.)

$x \in \mathbb{F} \cap [0, 2^{-26}) \implies \cdots$

$x \in \mathbb{F} \cap [2^{-26}, 2^{-2}) \implies$ Show $|\mathtt{p}(x) - \mathrm{atanh}(x)| \leq \frac{1}{2}\mathrm{ulp}(\mathrm{atanh}(x))$. **But how?**

$x \in \mathbb{F} \cap [2^{-2}, \infty) \implies \cdots$

Proof Overview

Example: Proof of binary64 `atanh` (ARITH 2026).

Subclaim: $|\mathrm{atanh}(x) - \mathtt{p}(x)| \leq \frac{1}{2}\mathrm{ulp}(\mathrm{atanh}(x))$.

Proof (simplified).

$$|\mathrm{atanh}(x) - \mathtt{p}(x)| \leq |\mathrm{atanh}(x) - p(x)| + |p(x) - \mathtt{p}(x)|$$

Proof Overview

Example: Proof of binary64 `atanh` (ARITH 2026).

Subclaim: $|\mathrm{atanh}(x) - \mathtt{p}(x)| \leq \frac{1}{2}\mathrm{ulp}(\mathrm{atanh}(x))$.

Proof (simplified).

$$
\begin{aligned}
|\mathrm{atanh}(x) - \mathtt{p}(x)| &\leq |\mathrm{atanh}(x) - p(x)| + |p(x) - \mathtt{p}(x)| \\
|\mathrm{atanh}(x) - p(x)| &\leq \varepsilon_{\mathrm{appr}}(x) \\
|p(x) - \mathtt{p}(x)| &\leq \varepsilon_{\mathrm{eval}}(x) \\
\varepsilon_{\mathrm{appr}}(x) + \varepsilon_{\mathrm{eval}}(x) &\text{ is sufficiently small} \\
&\ldots
\end{aligned}
$$

Proof Overview

Example: Proof of binary64 `atanh` (ARITH 2026).

Subclaim: $|\mathrm{atanh}(x) - \mathtt{p}(x)| \leq \frac{1}{2}\mathrm{ulp}(\mathrm{atanh}(x))$.

Proof (simplified).

$$
\begin{aligned}
|\mathrm{atanh}(x) - \mathtt{p}(x)| &\leq |\mathrm{atanh}(x) - p(x)| + |p(x) - \mathtt{p}(x)| \\
|\mathrm{atanh}(x) - p(x)| &\leq \varepsilon_{\mathrm{appr}}(x) && \leftarrow \textbf{Sollya} + \text{manual proof} \\
|p(x) - \mathtt{p}(x)| &\leq \varepsilon_{\mathrm{eval}}(x) && \leftarrow \textbf{Gappa} + \text{manual proof} \\
\varepsilon_{\mathrm{appr}}(x) + \varepsilon_{\mathrm{eval}}(x) &\text{ is sufficiently small} && \leftarrow \text{manual proof} \\
&\cdots && \leftarrow \text{manual proof}
\end{aligned}
$$

When the proof is too complex, use **exhaustive testing**.
(E.g., $\geq$ 1.5 years of CPU time on a 64-core machine for `atanh`.)

Correctness Issues Found

Previous error bounds used in "Ziv's rounding test" were **too small**.

Example: `acos`. For $0.062 \leq |x| \leq 1.000$,

prev. bound $= g(x) \cdot (2.87 \times 10^{-16})$;

sound bound $= g(x) \cdot (3.33 \times 10^{-16})$.

More examples: `acosh`, `asin`, `asinh`, `cosh`, `exp`, `expm1`, `sinh`, `tanh`.

Correctness Issues Found

Previous error bounds used in "Ziv's rounding test" were **too small**.

Example: `acos`. For $0.062 \leq |x| \leq 1.000$,

prev. bound $= g(x) \cdot (2.87 \times 10^{-16})$;

sound bound $= g(x) \cdot (3.33 \times 10^{-16})$.

More examples: `acosh`, `asin`, `asinh`, `cosh`, `exp`, `expm1`, `sinh`, `tanh`.

These unsound bounds resulted in **incorrect outputs**.

$$\texttt{sinh}(-5.014\ldots) = -1.2\texttt{d45dac997dc5}_{(16)} \times 2^6$$
$$\mathrm{rnd}(\sinh(-5.014\ldots)) = -1.2\texttt{d45dac997dc4}_{(16)} \times 2^6,$$

$$\texttt{acos}(-0.498\ldots) = +1.0\texttt{be9b1d192aca}_{(16)} \times 2^1$$
$$\mathrm{rnd}(\arccos(-0.498\ldots)) = +1.0\texttt{be9b1d192acb}_{(16)} \times 2^1, \text{ and many more.}$$

From Pen-And-Paper to Formal Proofs

We are **formally proving** parts of some CORE-MATH functions.

Members: Wonyeol, Hyunsoo, Hyeonjung, Hyunwoo (with help from Paul).

Functions: binary64 `atanh` and `exp`.

Approach: Formalize pen-and-paper proofs in Rocq.
Use `interval` and `gappa` tactics for Sollya and Gappa parts.

From Pen-And-Paper to Formal Proofs

We are **formally proving** parts of some CORE-MATH functions.

- Members: Wonyeol, Hyunsoo, Hyeonjung, Hyunwoo (with help from Paul).
- Functions: binary64 `atanh` and `exp`.
- Approach: Formalize pen-and-paper proofs in Rocq.
  Use `interval` and `gappa` tactics for Sollya and Gappa parts.

Goal for 2027: Formally prove a **complete** binary64 function.

- Good candidate: binary64 `exp`.
- Preliminary studies: Exhaustive testing may not be necessary.
- Hope: Present the results at FPTalks 2027?

Take-Home Messages

To gain confidence in floating-point code, we need **pen-and-paper/formal proofs**.

Proving the CORE-MATH functions helped us to identify **subtle correctness issues**.

Please contact us if you want to join the project!