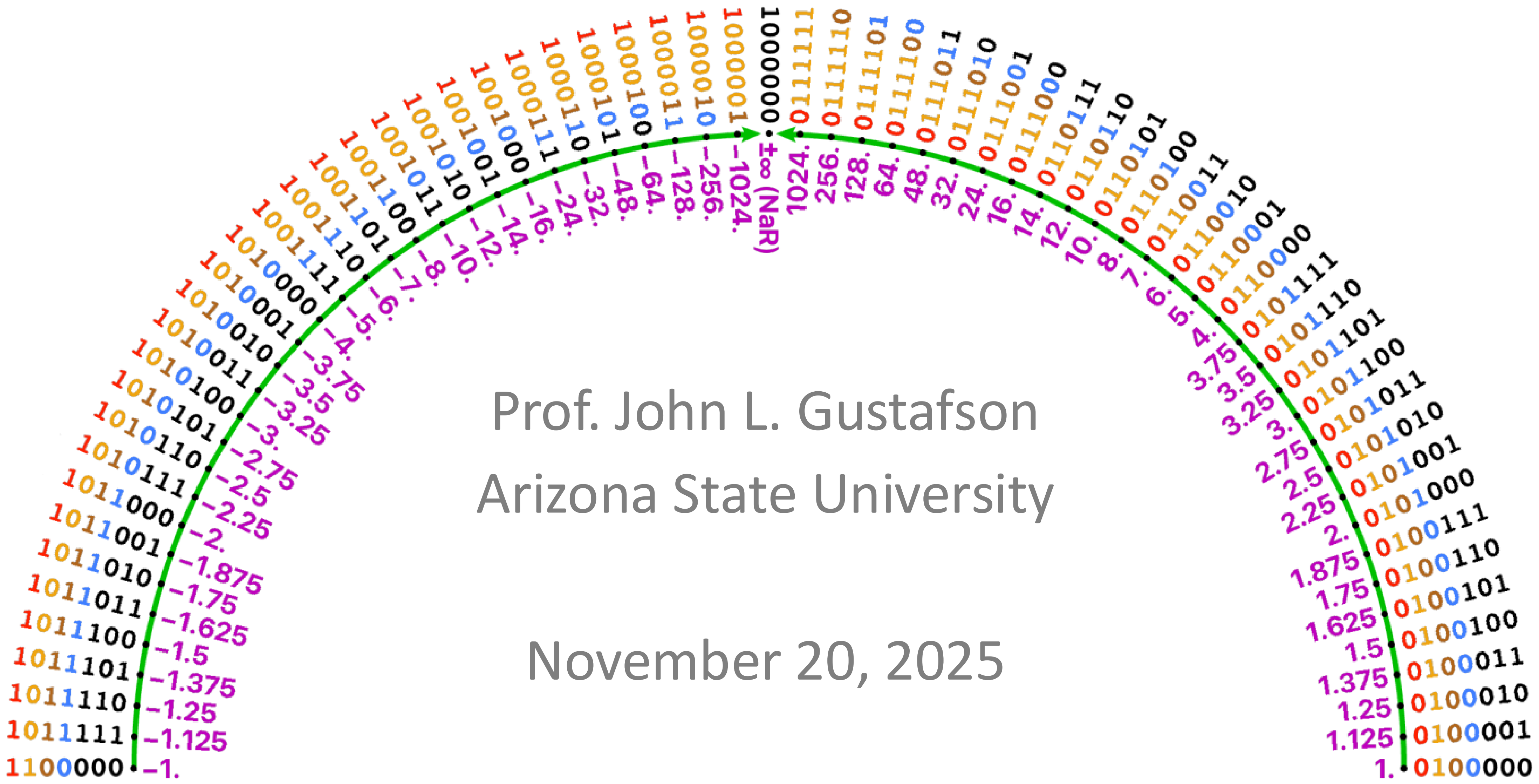
Welcome To CoNGA 2025

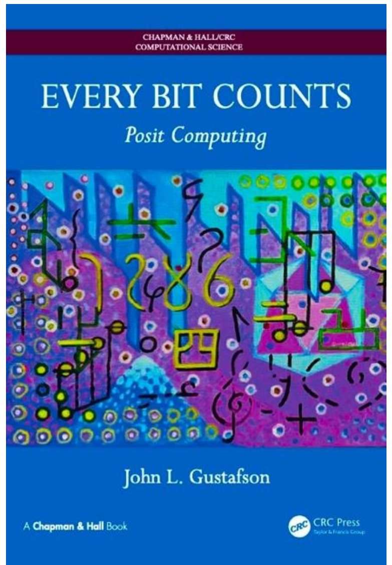


Prof. John L. Gustafson
Arizona State University

November 20, 2025

Since the last CoNGA (February 2024)

- **CoNGA now moves from SCAsia in February to SC in November (thanks to Elizabeth Leake!) and Kurt Keville now co-chairs.**
- **Calligo Tech is shipping RISC-V systems with native posits in VLSI.**
- ***Every Bit Counts: Posit Arithmetic* published in December 2024.**
    - **Defines 11 engineering goals for a format.**
    - **Covers "sane floats", takums, b-posits, logarithmic number systems, biased exponents, asymmetric tapering, and full custom systems.**



First to market. First to scale.

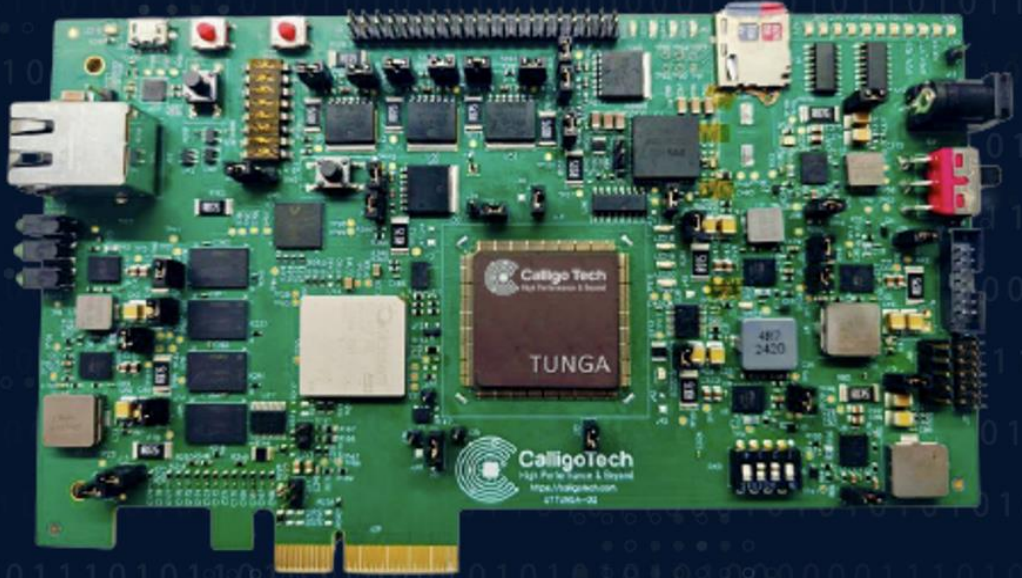
World's first Posit-enabled ASIC – TUNGA 1.0

Powering Accelerator card – UTTUNGA 1.0

Built by Calligo Technologies, a deep-tech pioneer redefining global computing by bringing POSIT arithmetic to silicon!

Integrates with legacy libraries **(BLAS, MAGMA)** without source changes

Compatible with any **PCIe-based x86/ARM server**



Calligo Tech
High Performance & Beyond

EXTENDED CAPABILITES

QUIRE | Custom Kernels

APPLICATION LAYER

GROMACS FAST. FLEXIBLE. FREE. | OpenFOAM | CP2K

LIBRARIES/FRAMEWORKS

OPEN MPI | ONNX RUNTIME | hypre | OpenMP | PyTorch | TensorFlow Lite | OpenBLAS

COMPILERS

LLVM | MLIR | GLOW | tvm

PROFILER & DEBUGGER

OpenOCD | Debug

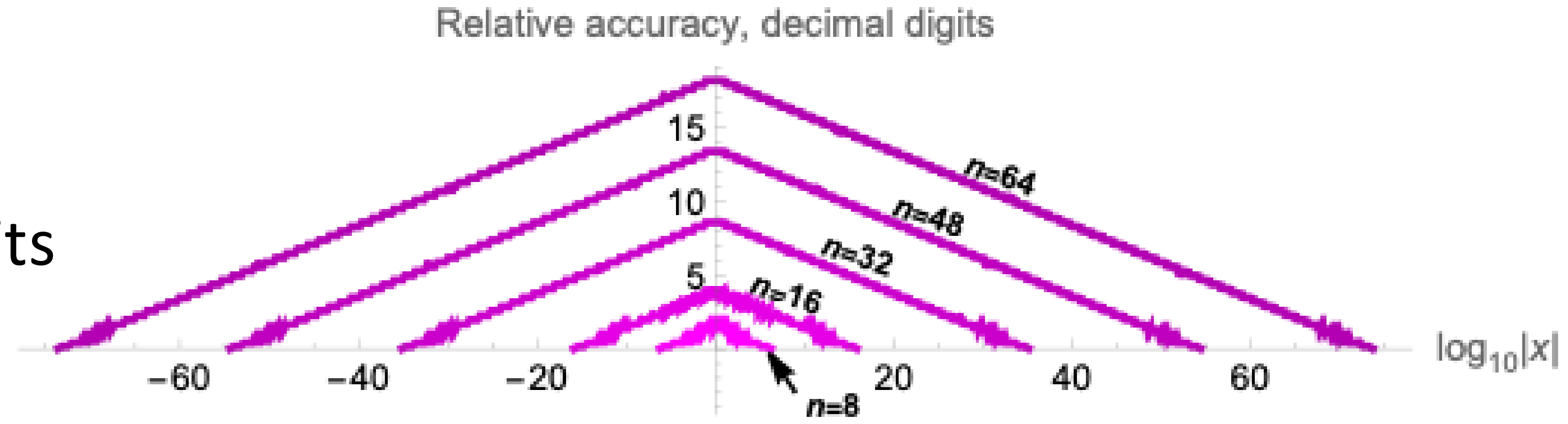
Ease of Use & Co-Exist

Flexible Use - Either compile with POSIT compiler or run existing code without change.
Seamless Integration - Offloads only computations to the card, results sent back to host
Co-Exists - with all other compute platforms CPU, GPU, NPU, DPU Etc... .

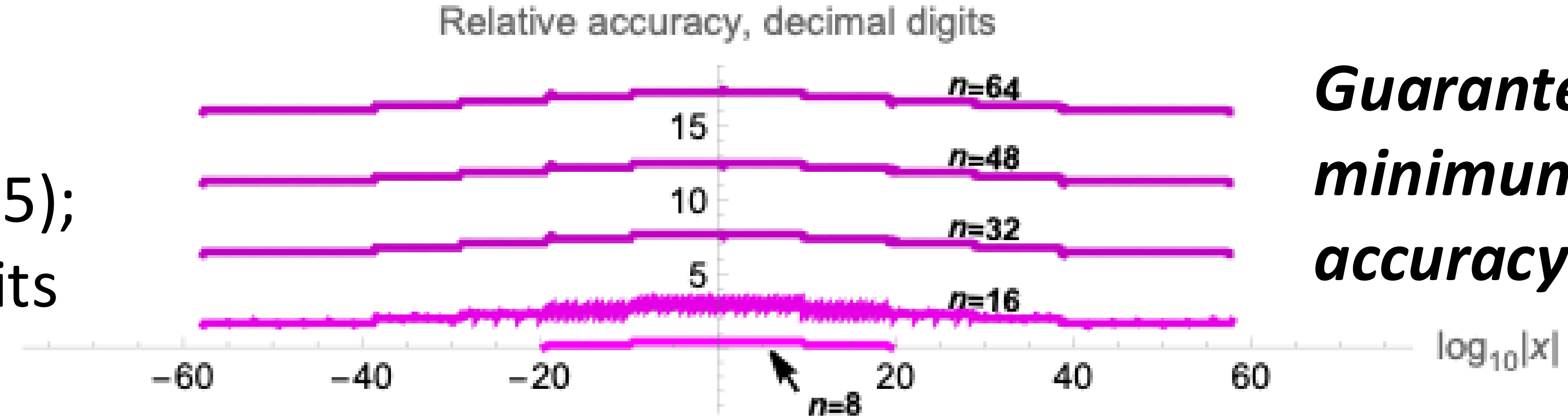
Since the last CoNGA (cont.)

The *b-posit* (bounded posit) fixes posit shortcomings pointed out by Florent de Dinechin, and beats floats for hardware-friendliness.

standard posits;
quire takes 16*n* bits



b-posit (*rS*=6, *eS*=5);
quire takes 800 bits



Guaranteed minimum accuracy

11 Design Principles for NGA

1. Distribution of representable values closely resembles the distribution needed for exact calculations.
2. No redundant bit patterns. Every bit counts.
3. No hidden "modes" of operation. Source program and data suffice to determine every bit of the output.
4. *All* math operations must be correctly rounded.
5. It should be easy to change precisions, like it is for integers.
6. Error results must always propagate through to the final output.

11 Design Principles for NGA (cont.)

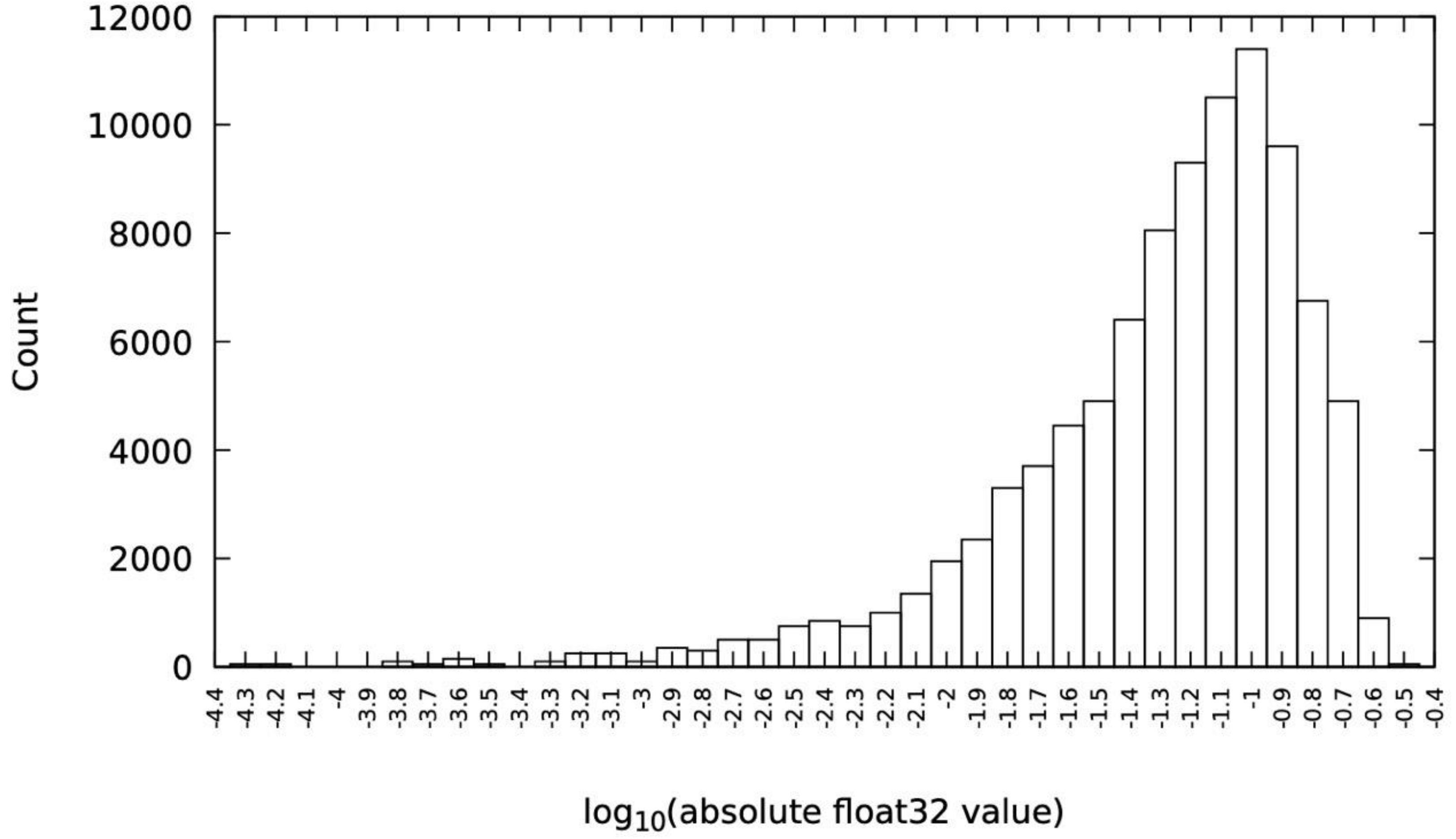
7. It should be simple to build in hardware, and fast.

In the "nice to have" category:

8. It should be easy to scale to any number of bits.
9. Real numbers should be ordered like integers with the same bit strings. (No extra hardware for comparison operations).
10. Negating a real should be the same as negating it as an integer.
11. The results of application computations should be close to what they would be if infinite precision is used.

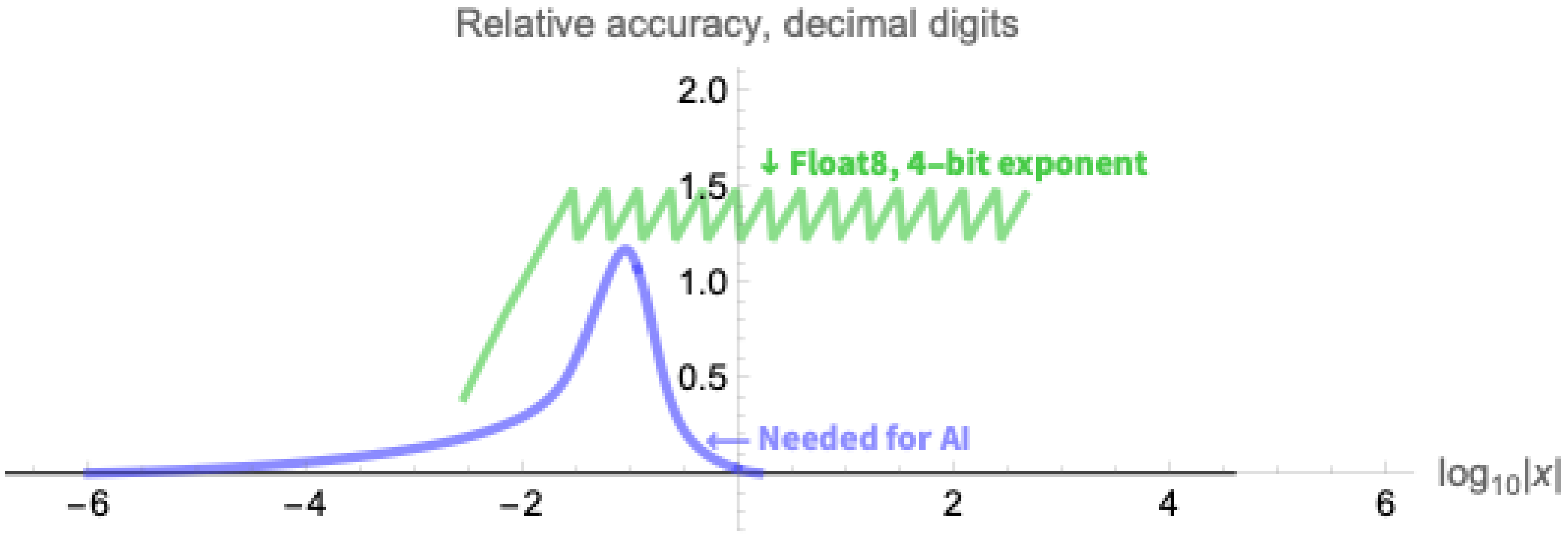
Note: IEEE Std 754 floats fail ***every one of these goals*.**

Typical distribution of values needed for AI



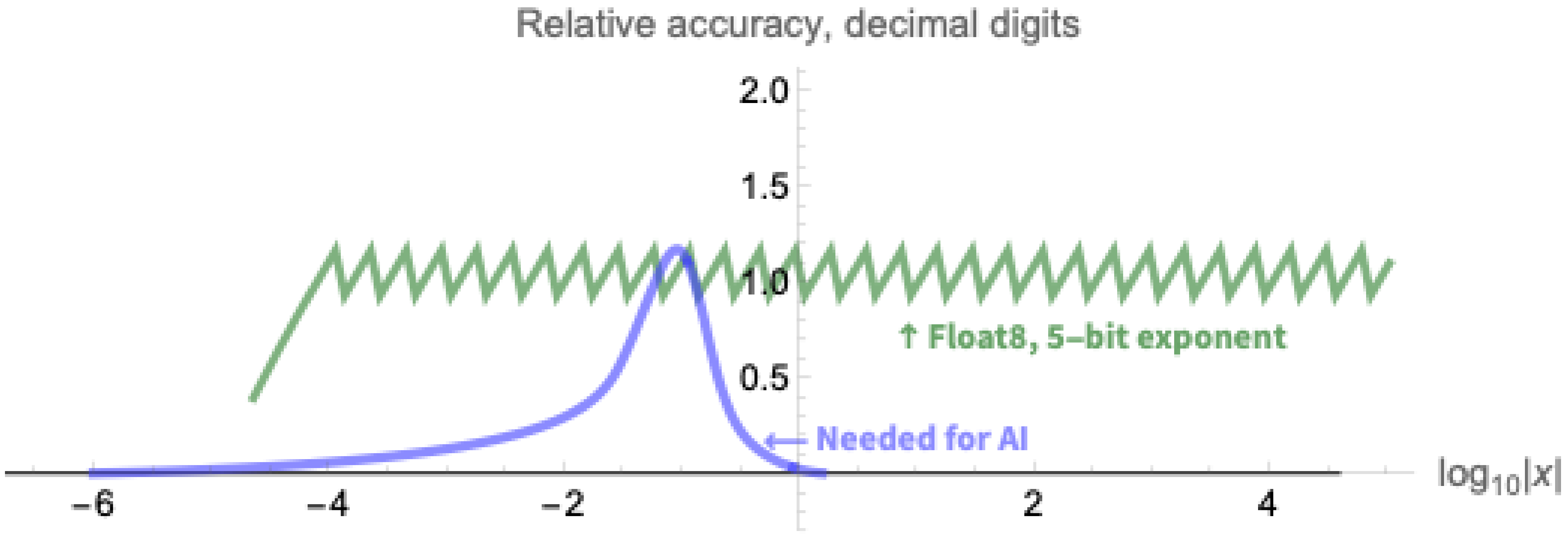
Generalized posits can match this distribution almost perfectly.

8-bit floats with 4 exp. bits mismatch AI needs



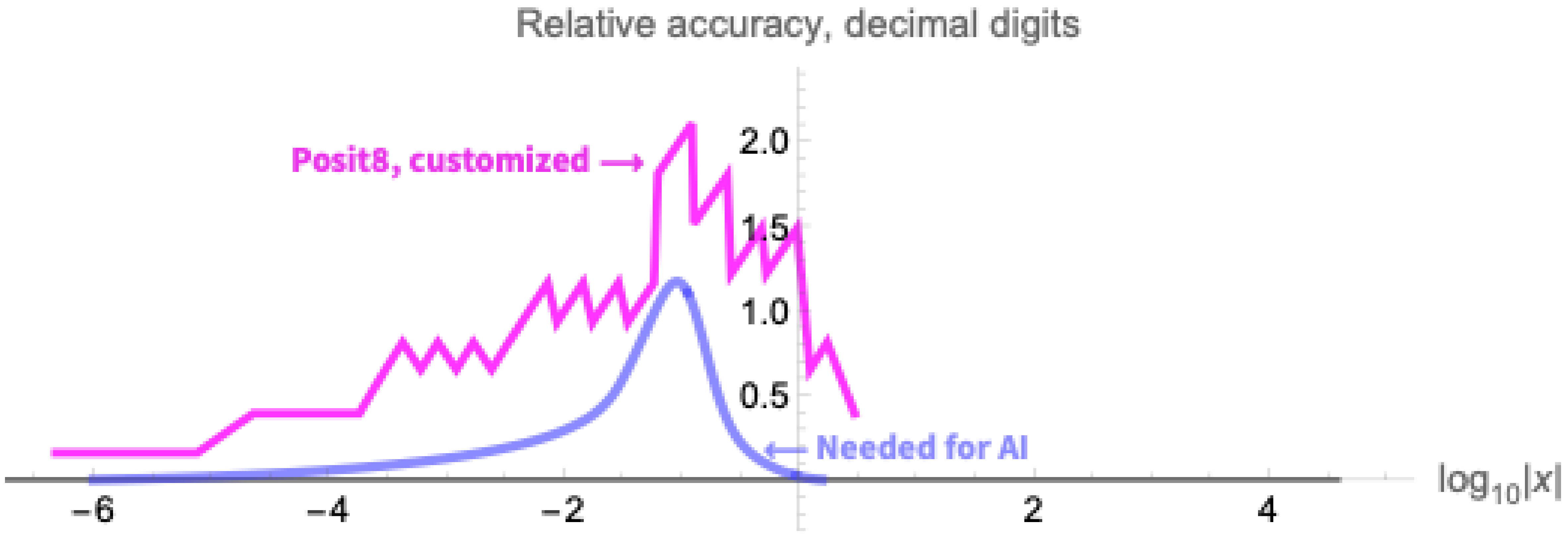
Insufficient dynamic range, even if you shift left by scaling.

FP8 exponent bits to 5 for more range



Insufficient accuracy where it is most needed.

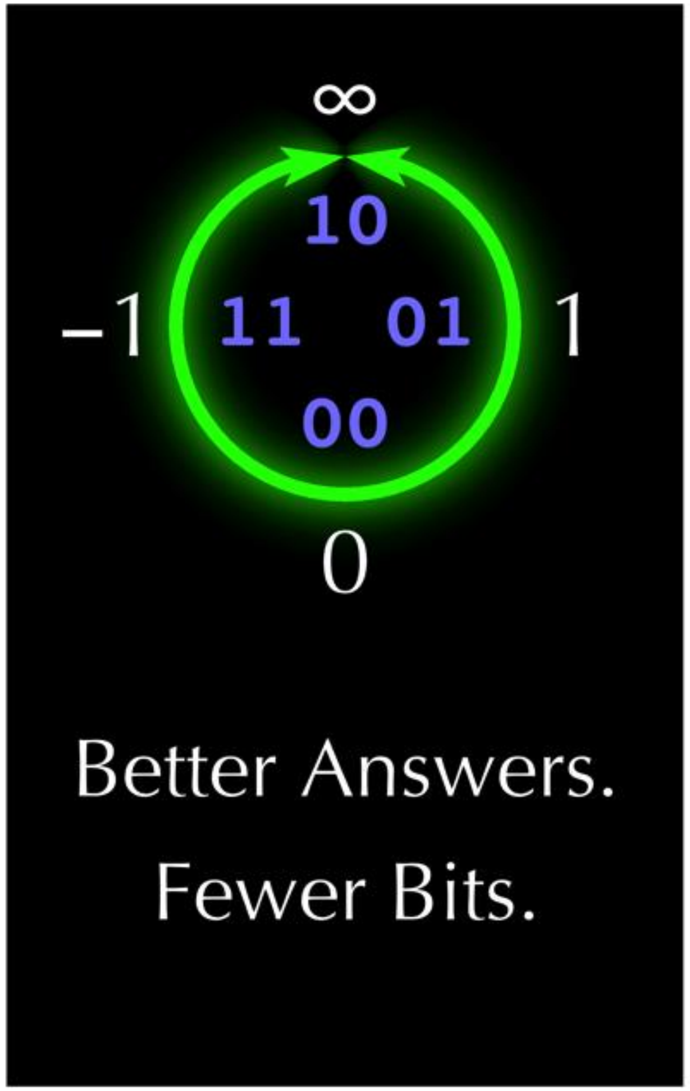
Posit8 distribution can be customized.



Near-perfect fit for both training and inference. Hardware-friendly.

Future Directions for NGA

- Curate real number distributions needed for the entire range of HPC apps and AI.
- Find extensions to languages like C and C++ that allow NGA and floats to coexist
- Use recent hardware breakthroughs to build NGA cores that are faster, smaller, and lower energy per op than float-based cores *at the same precision.*


∞
10
−1
11
01
1
00
0
Better Answers.
Fewer Bits.