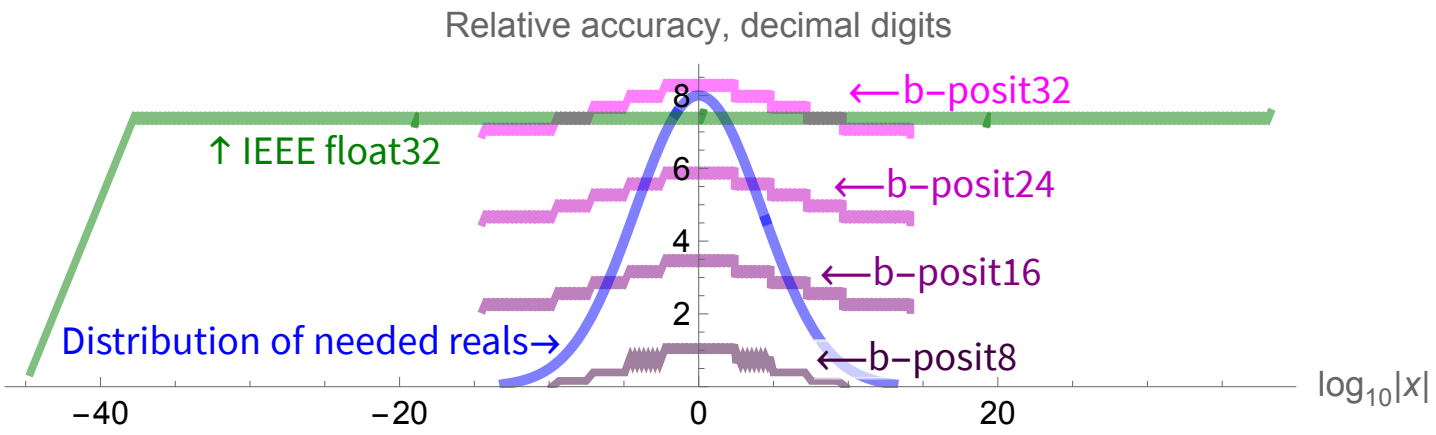
Variable Precision Formats that Preserve Dynamic Range



Prof. John L. Gustafson
Visiting Scholar / Researcher
Arizona State University

HPEC 2026
September 18, 2026

Thesis

Changing real-number precision should be fast and easy.

With IEEE 754 floats, it is slow and difficult (decode+re-encode).

Two recent inventions, the *b-posit* and the *takum* format, get to the necessary dynamic range with the lowest precision possible, then *only add fraction bits for more accuracy*.

This makes changing precision as easy for real numbers as it is for integers.

Energy cost of a 64-bit IEEE float multiplication, in 2026:

Task	Energy, nanojoules
All data in registers	0.002 to 0.02
All data in L1 or L2 cache	0.1 to 0.4
All data in L3 cache	0.2 to 1.
All data in HBM3 or HBM4 DRAM (like GPU stacks)	0.6 to 1.5
All data in DDR5 DRAM (like server memory)	3 to 5.
All data in a different cabinet in a cluster	~20.

The multiplication operation itself	**0.005**

Data motion is 10× to 100× the energy cost of *arithmetic*.

Obvious conclusion: **Use fewer bits to get the work done.**

One argument for sufficiency of dynamic range

Extreme magnitude constant	Known value	32–bit posit form
Speed of light (physics)	2.99792458×10^{8}	$2.99792_{4}\times10^{8}$
Charge of an electron (electromagnetism)	$1.602176634\times10^{-19}$	$1.602_{2}\times10^{-19}$
Avogadro's number (chemistry)	6.02214076×10^{23}	$6.02_{1}\times10^{23}$
Boltzmann constant (thermodynamics)	8.31446262×10^{-23}	$8.31_{3}\times10^{-23}$
Planck's constant (quantum mechanics)	6.62607015×10^{-34}	$7.\times10^{-34}$
Cosmological constant (general relativity)	1.1056×10^{-52}	$7.\times10^{-34}$
Mass of known universe (cosmology)	1.5×10^{53}	1.3×10^{36}

Physical constants in increasing order of how extreme their magnitude is

Why IEEE 754 Precision-Changing is a Mess

Test the exponent bits for all 0 bits or all 1 bits…

6% of bit patterns

Trap to **software or microcode**, ~200 clocks!

$$\text{Value} = \begin{cases} \text{if } E = 31, \begin{cases} \text{if } F = 0, (-1)^S \times \infty \\ \text{else}, \begin{cases} \text{if } F \geq 0.5, \text{Quiet NaN} \\ \text{else, Signaling NaN} \end{cases} \end{cases} \\ \text{else if } E = 0, \begin{cases} \text{if } F \neq 0, \text{Subnormal}, (-1)^S \times 2^{E-14} \times (0 + F) \\ \text{else}, \begin{cases} \text{if } S = 0, \text{“Positive zero”} \\ \text{else, “Negative zero”} \end{cases} \end{cases} \\ \text{Else, a “normal” float. } (-1)^S \times 2^{E-15} \times (1 + F) \end{cases}$$

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.)

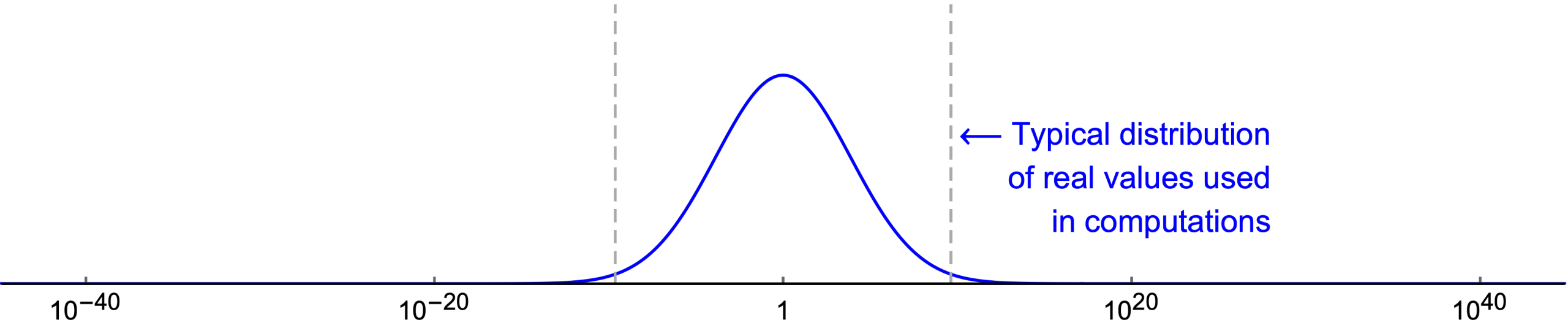
7. It should be simple to build in hardware, and fast.

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.

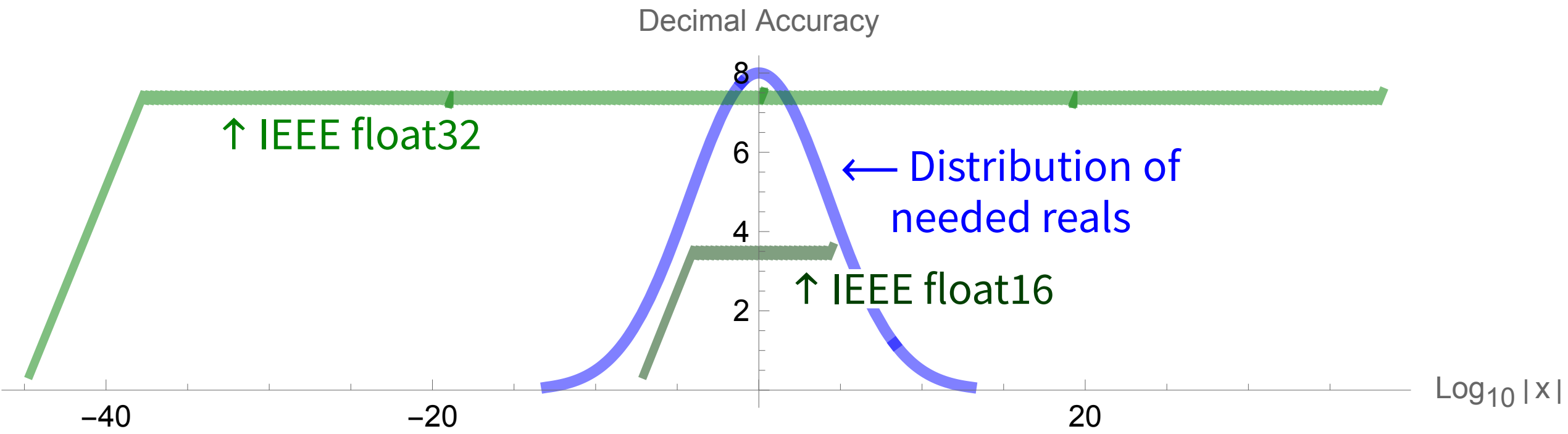
Note: IEEE Std 754 floats fail ***every one of these goals***.

What reals should we seek to represent?



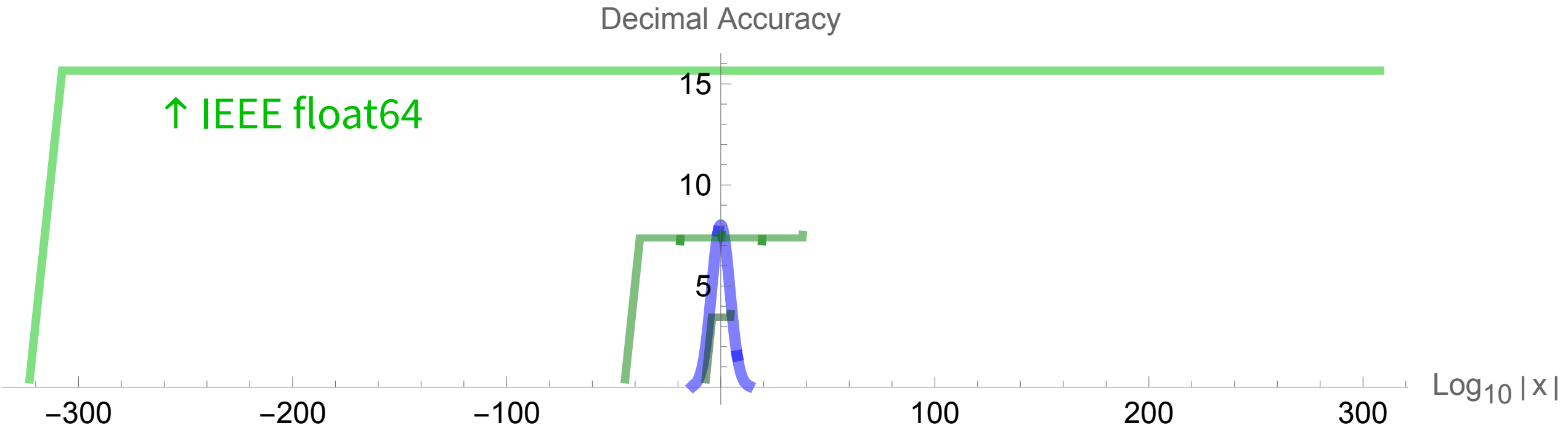
Studies show *very* rare use of values outside 10^{-13} to 10^{13}.
Central Limit Theorem says exponents distribute as a bell curve.

IEEE float16 too small, float32 often too large



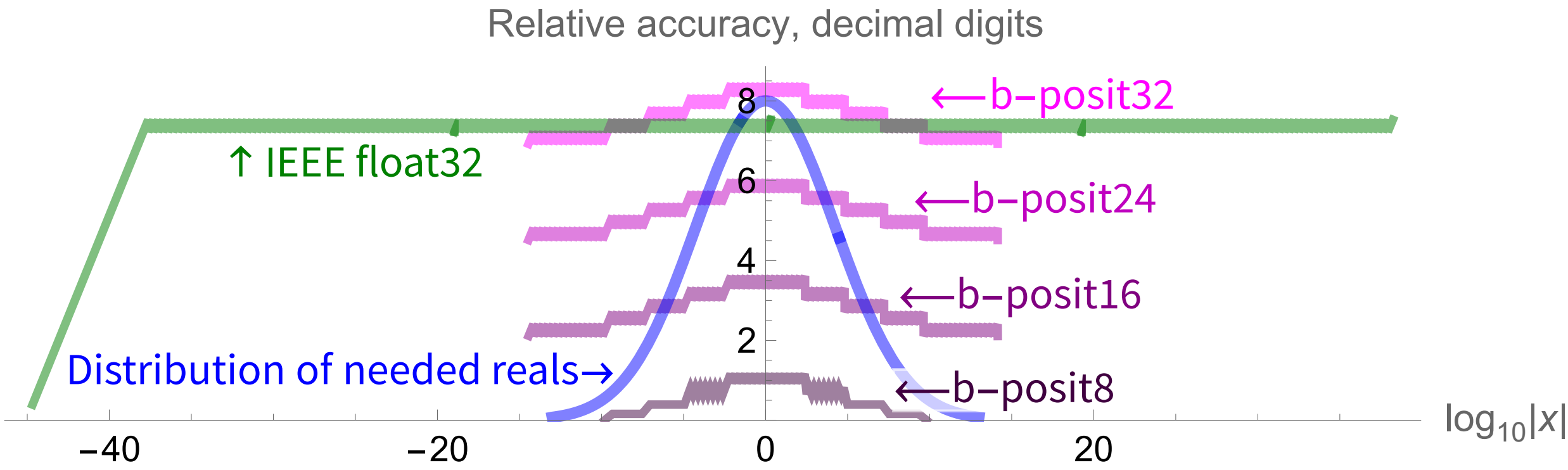
The 5-bit exponent of float16 is too small for almost everything.
The 8-bit exponent of float32 is often overkill.

IEEE float64 is *insanely* wasteful



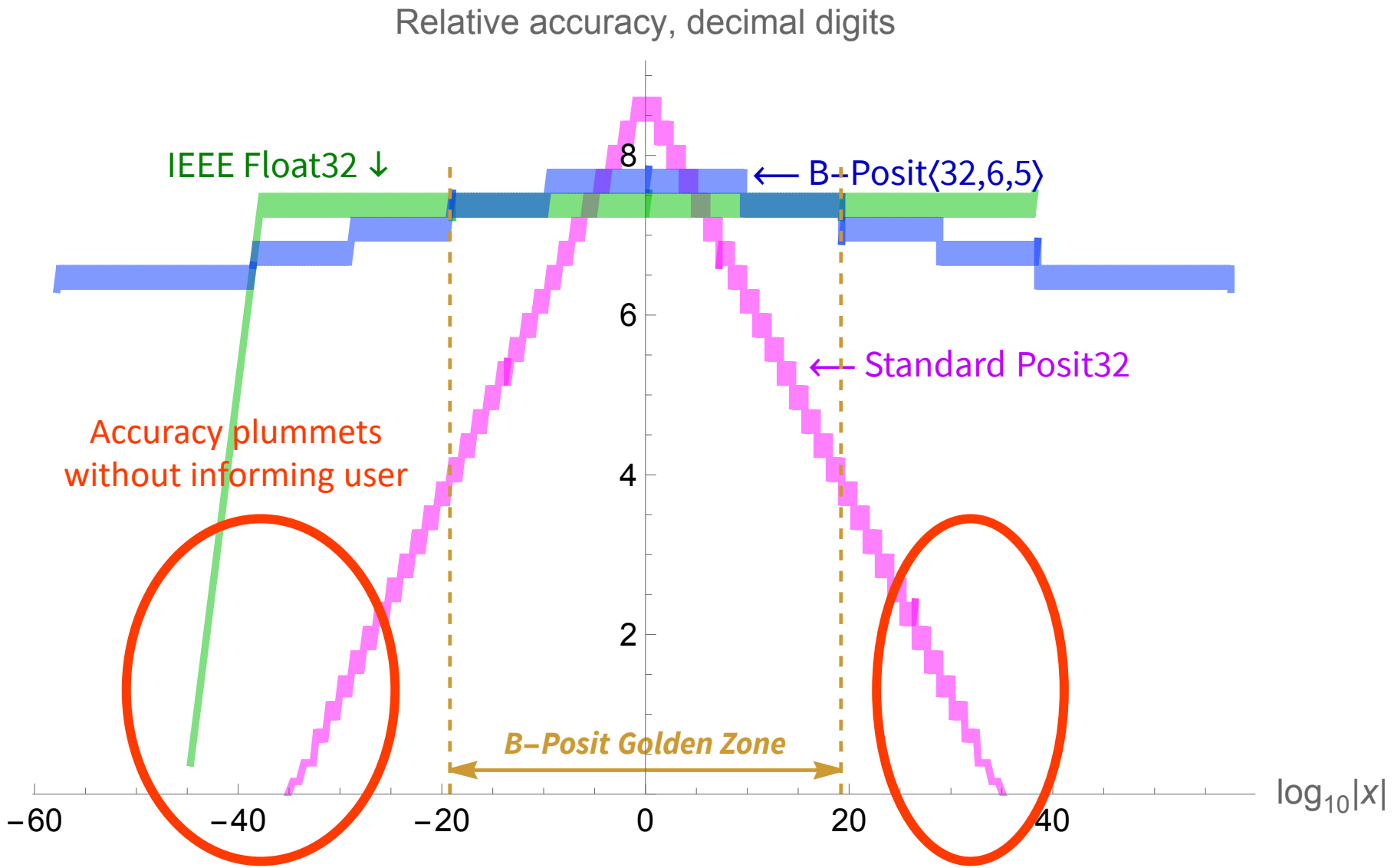
The 11-bit exponent stems from the desire (in 1977) to reduce transistor count in the integer multiplier for the significand.

Bounded posits (b-posits) fit what is needed



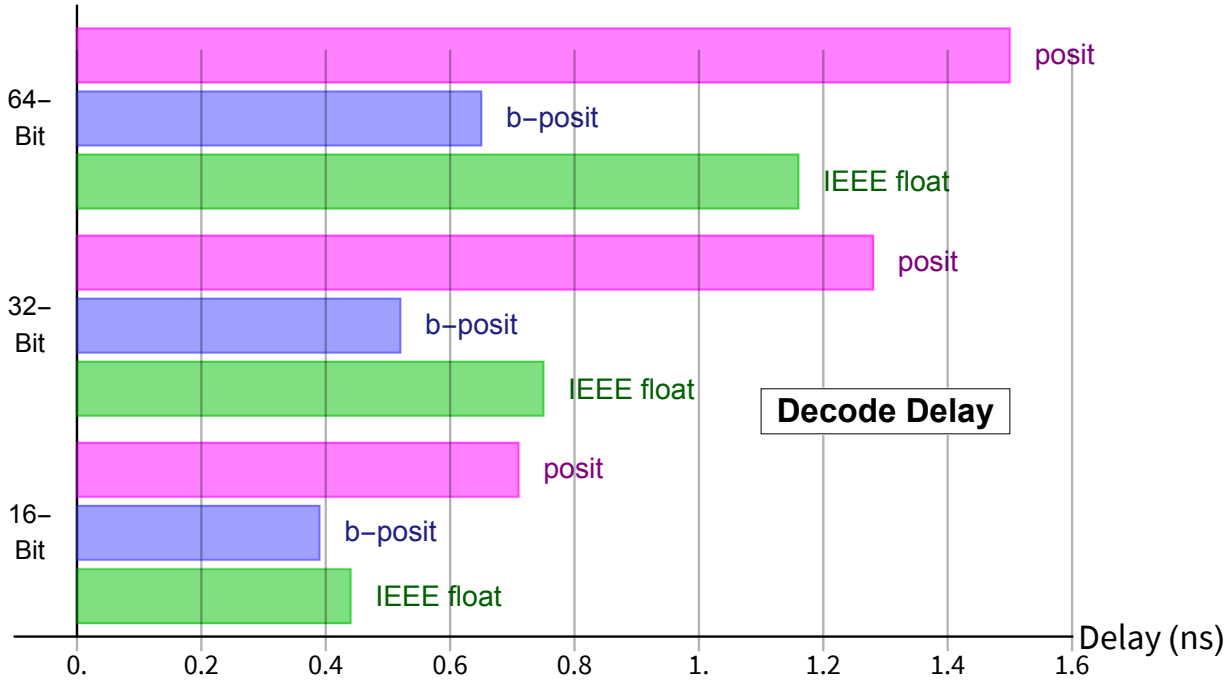
Changing precisions does not require decoding bit fields!
Numerical analysis bounds work for b-posits but not IEEE floats.

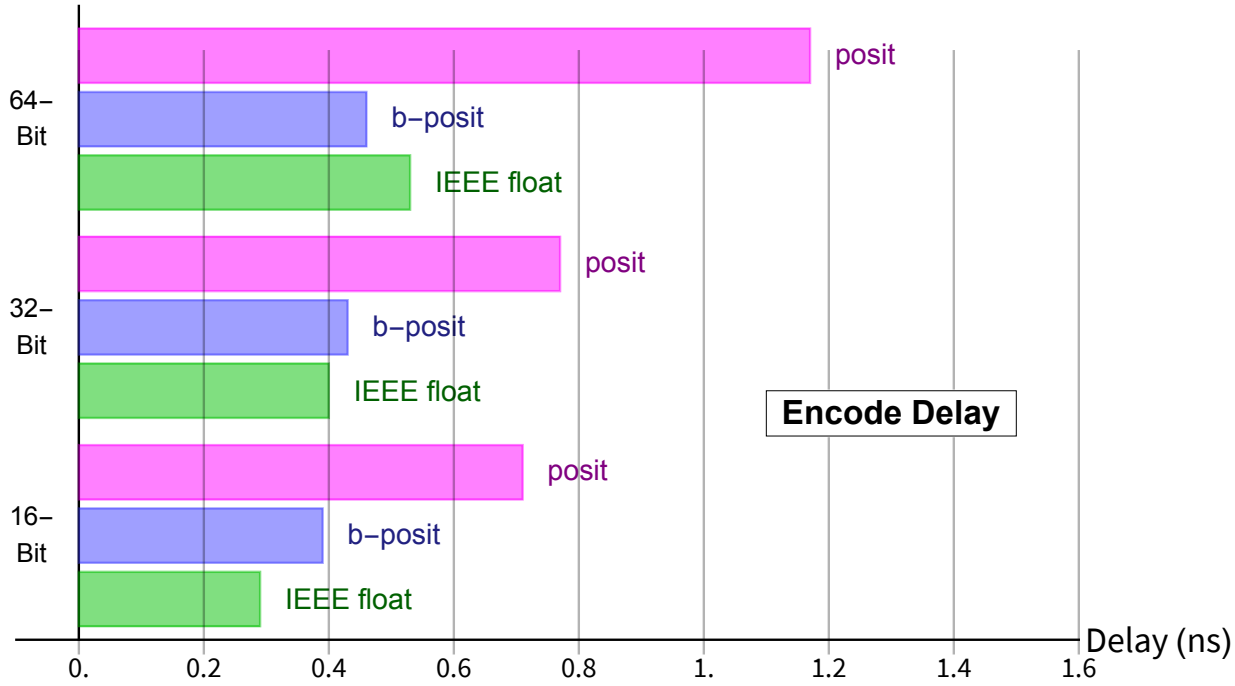
B-Posits for General Scientific Computing Range



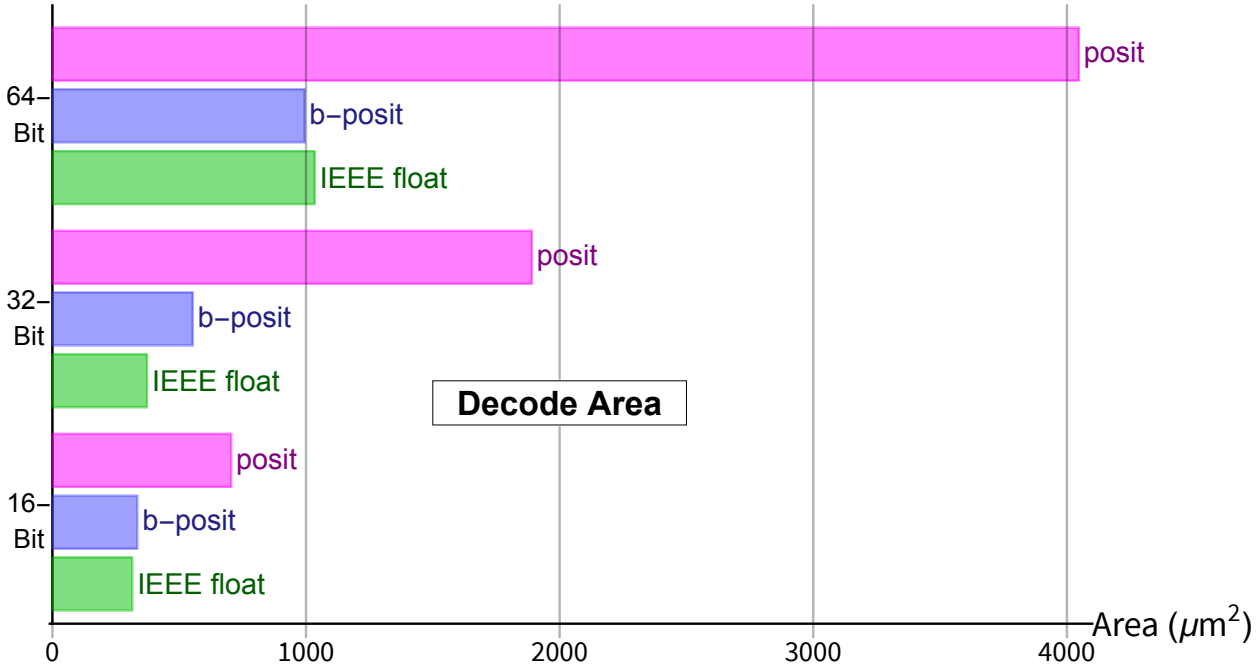
- Limit the maximum regime size
- Superior dynamic range
- Superior accuracy over vast range (Golden Zone)
- Greatly simplified hardware
- Lower latency

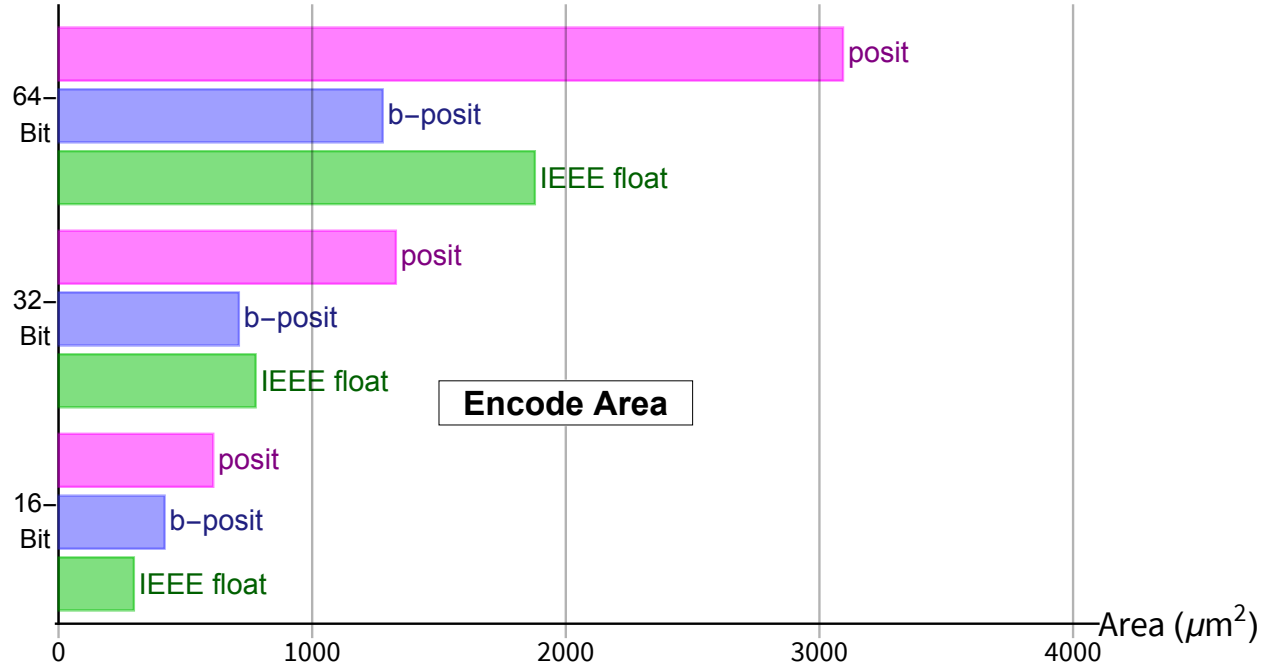
Worst-Case Decode-Encode Time





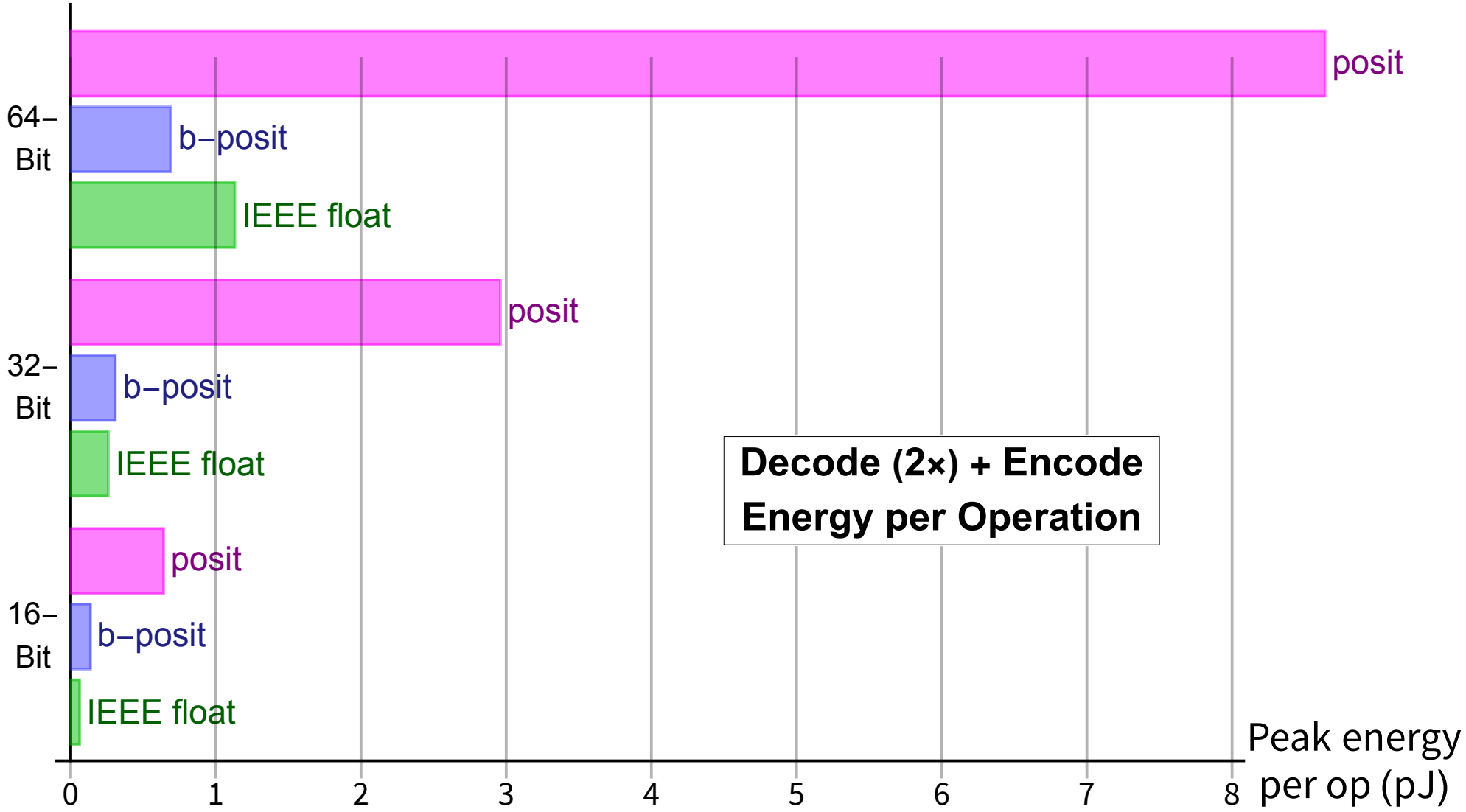
Decode-Encode Chip Area for Same Precision



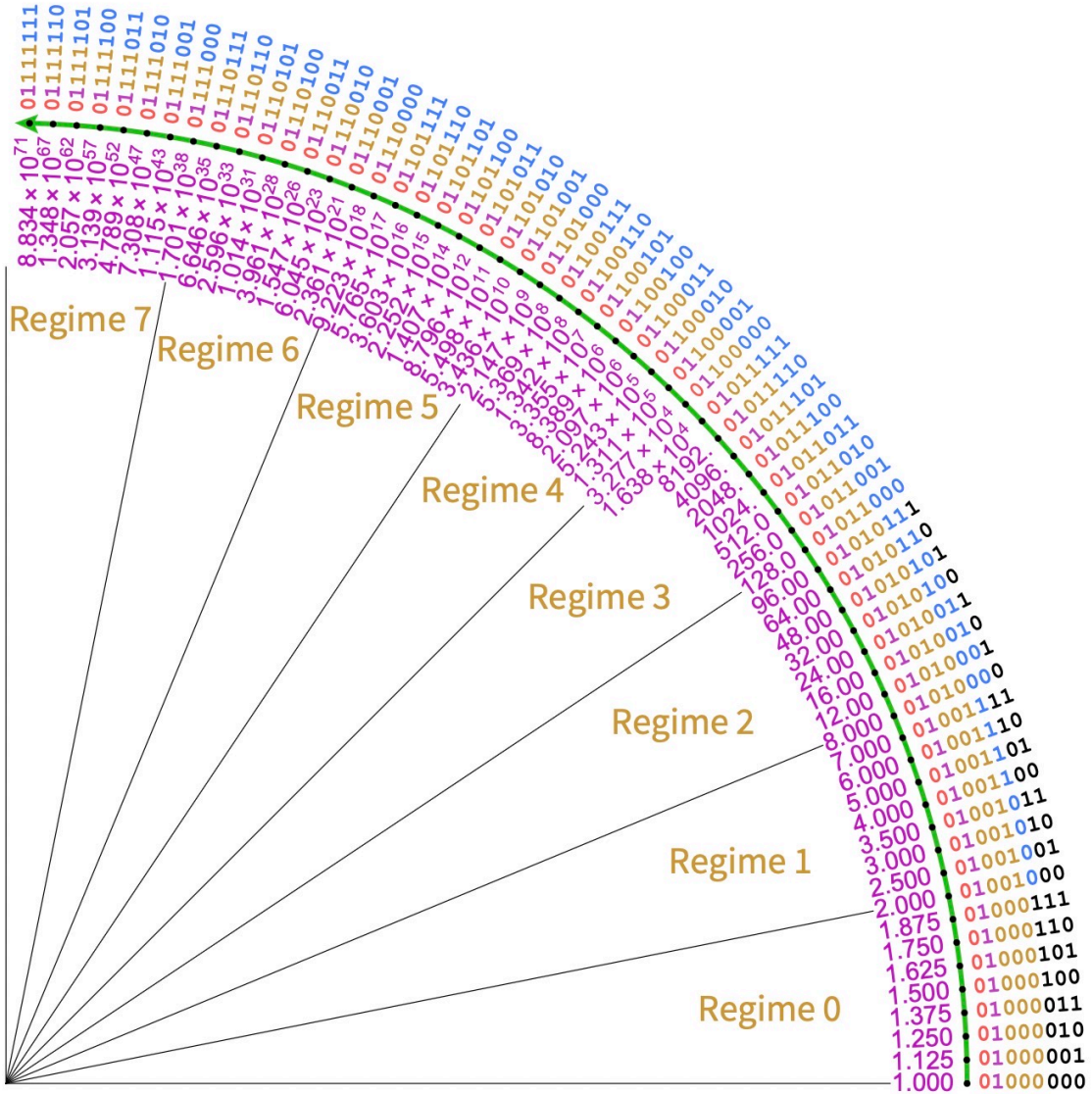


These are early results. More improvements likely in the future.

Worst-Case Power Consumption



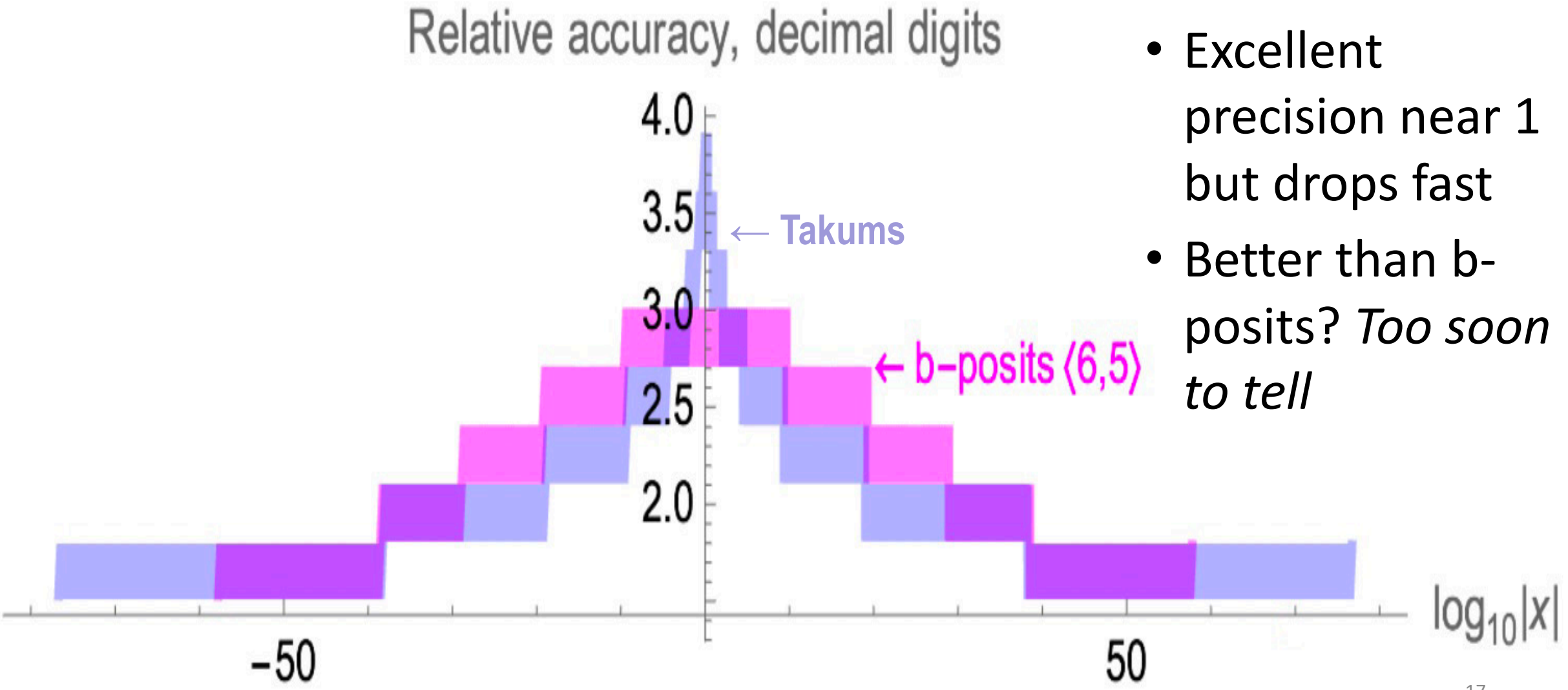
Takum format (L Hunhold) also bounds range



Upper right quadrant of a posit ring for takum format

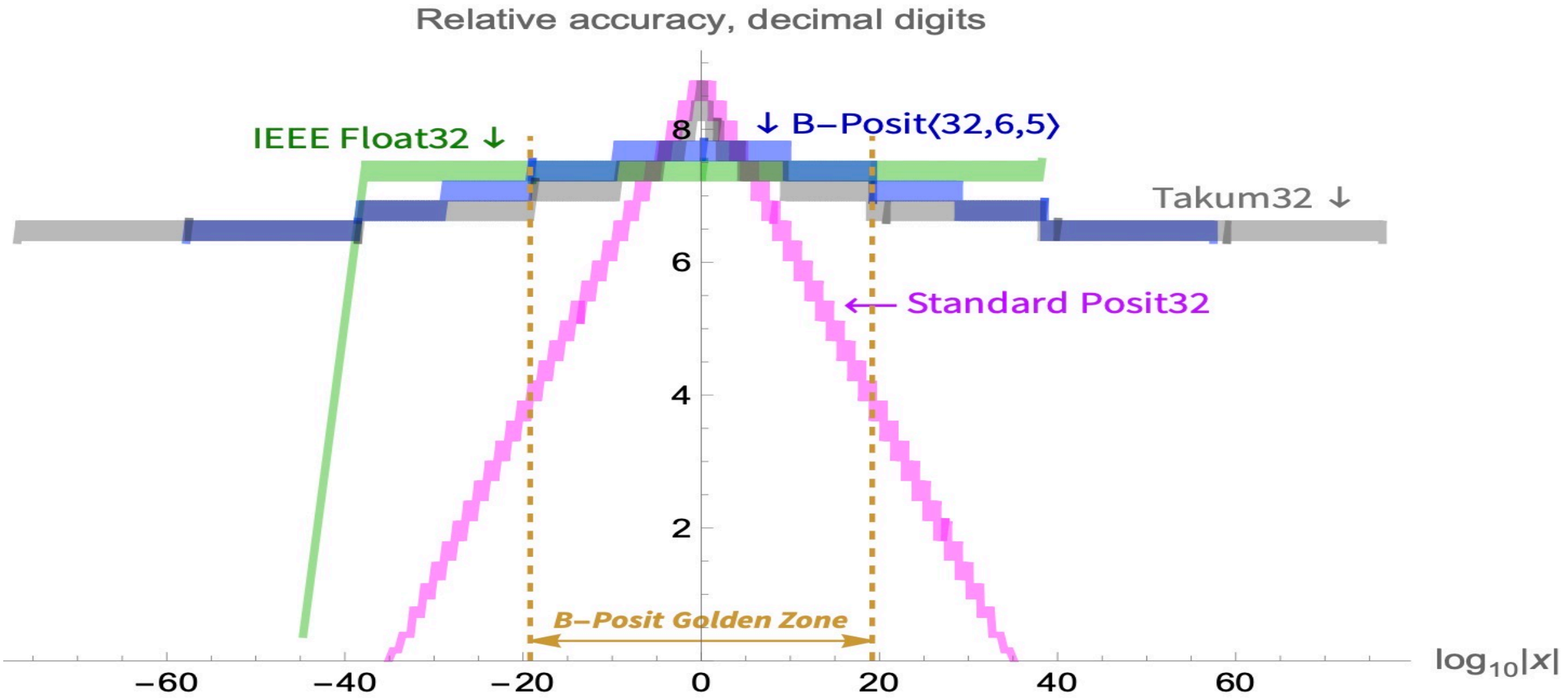
- Similarly allows fast MUX decoding
- Scaling 2^{-255} to 2^{255}
- Scaling encoded in four fields instead of three
- Fits the posit framework & design goals

16-bit takums and b-posits compared

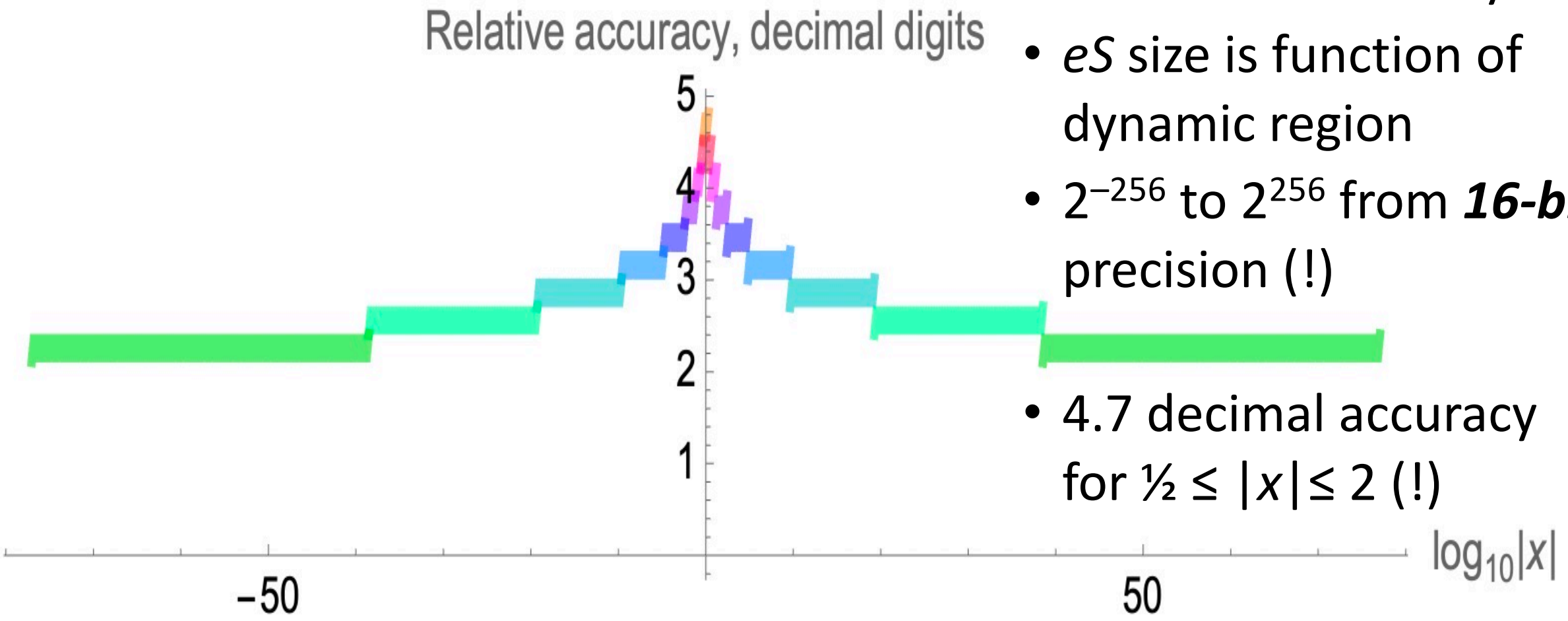


- Excellent precision near 1 but drops fast
- Better than b-posits? *Too soon to tell*

Four 32-bit formats compared



Breaking News… The Best of Both Worlds?



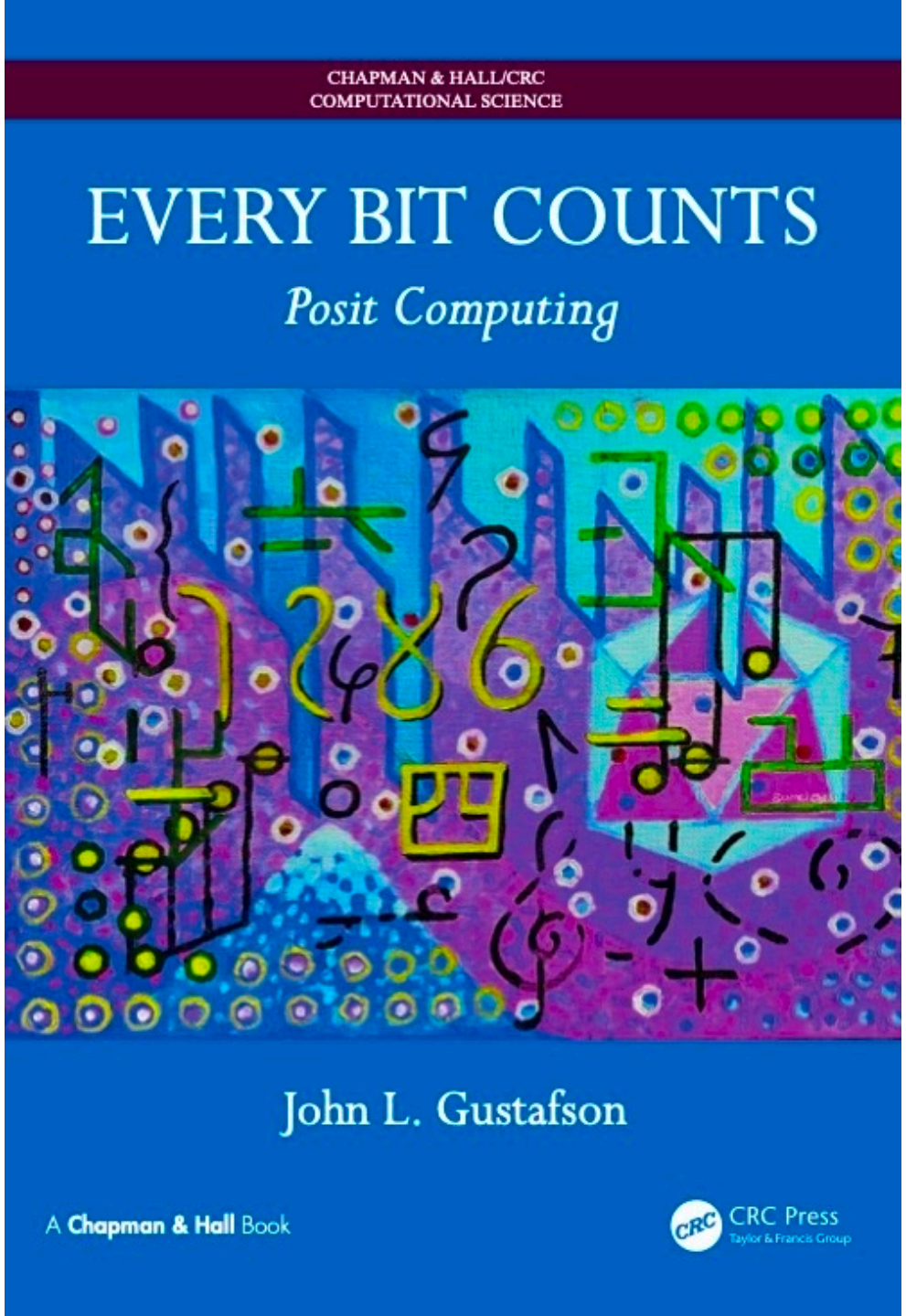
- Week-old discovery
- *eS* size is function of dynamic region
- 2^{-256} to 2^{256} from ***16-bit*** precision (!)
- 4.7 decimal accuracy for ½ ≤ |*x*|≤ 2 (!)

Shameless Plug…

Includes b-posit and takum formats.

464 pages, all in full color

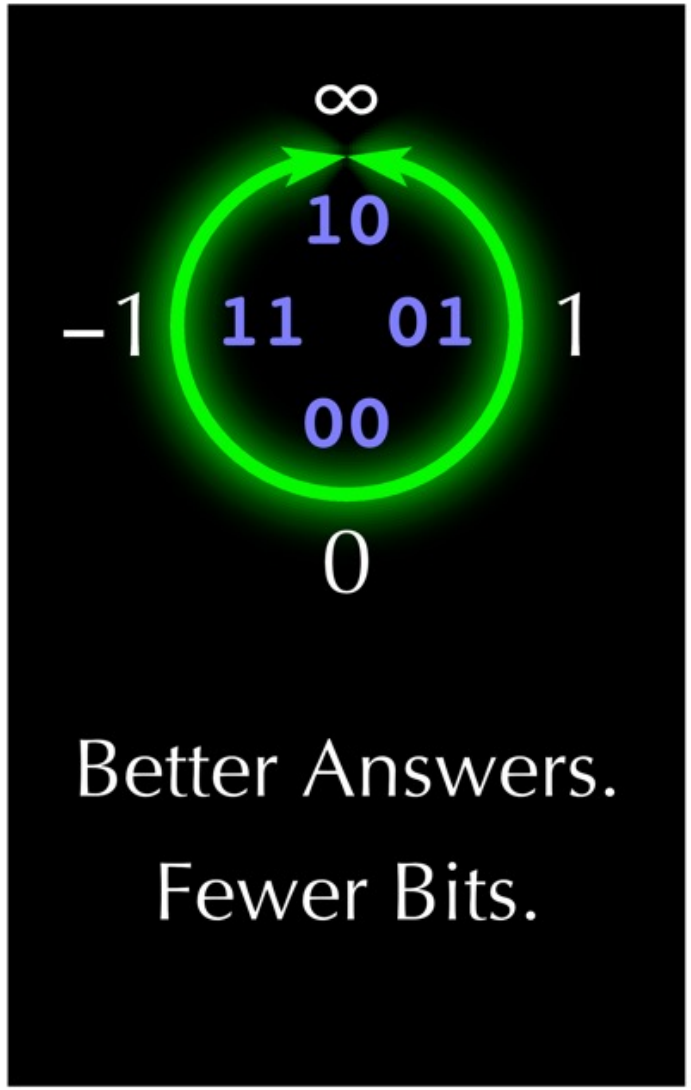
Shows how to customize real number formats to any workload, including AI.



Summary

- B-posits give a dynamic range 2^{-192} to 2^{192} (about 10^{-58} to 10^{58} for precision ≥12 bits)
- Takums use a similar method to go from 2^{-255} to 2^{255} (about 10^{-77} to 10^{77} for precision ≥12 bits)
- Both make mixed precision very, very easy.
- Both can guarantee minimum relative accuracy, unlike posits and IEEE floats

Thank you!
For more information on posits:
www.posithub.org




TESLA
Roadster Sport
ARIZONA
POSIT