Errorless CPUs for Better AI and Compute

James Douglas Boyd
Founder and CEO/CTO, SciSci Research

Exact, Errorless Computing

SciSci Research is building the world's first errorless CPU, capable of performing exact arithmetic without rounding error or approximation. Since the 1940's, CPUs have performed arithmetic on floating-point numbers ("floats"), i.e., finite decimal approximations of numbers, and thus suffer from rounding errors. Replacing this paradigm, exact CPUs will deliver better results for AI and scientific computing with perfect precision and scalable compute.

Exact Computing for Better AI

Exact computing will make AI results reproducible and precise. Currently, LLM results are not reproducible; results are inconsistent, even with deterministic sampling. Thinking Machines identifies floating-point as the underlying cause: due to rounding, outputs from the same computation vary depending on the order of operations. As for LLM precision, losses occur, for instance, during post-training quantization, which approximates parameters. Exact CPUs will fix these problems and imprecision issues generally.

Exact computing is also more efficient and scalable. NVIDIA, IBM, and others scale floating-point AI compute via "low-precision" computing, due to the inefficiency of floats with many decimals. Floating-point arithmetic is itself inefficient, involving steps such as rounding, comparisons, and shifts, due to approximation and the somewhat complicated structure of the IEEE floating-point standard. By contrast, exact arithmetic does not require rounding, comparisons, or shifts; the CPU just adds the numbers. Arithmetic is efficient, instructions are simpler, and pipelining will be more predictable, all favorable for scaling.

Exactness for Scientific Computing

Even for applications that are not using low-precision computing, such as scientific computing, exact computing will be crucial for overcoming the errors inherent to floating-point. The IEEE floating-point standard, in the case of double-precision, leaves a rounding error of $\approx 10^{-16}$. Compare this with the "accelerated computing" GPU frontier, which is over the petascale threshold, performing $> 10^{16}$ floating-point operations per second (FLOPS). Even if double-precision can achieve this performance benchmark, the error will be > 1 per second. At greater scales (e.g., zeta-), errors will drastically accumulate. By contrast, exact computing will deliver perfect precision to models and simulations in drug-discovery, manufacturing, defense, finance, and other scientific computing applications.

How Exact Computing Works

Exact arithmetic, although never before realized in hardware, already exists in software, namely in packages written by number theorists for computer algebra systems such as Magma. Exact arithmetic itself comes from number theory. Floats are real numbers (i.e., in $\mathbb{R}$), approximated as finite decimal expansions by rationals (i.e., in $\mathbb{Q}$). Real numbers "complete" rationals, giving them convergent infinite sequences (Cauchy sequences). The issue is that in $\mathbb{R}$, these sequences aren't unique, and suffer encoding collisions (rounding errors) under approximation. However, the p-adic fields $\mathbb{Q}_p$ (with p a fixed prime) are also completions of $\mathbb{Q}$ and have unique Cauchy sequences, admitting finite encodings amenable to exact arithmetic. Rather than use floats, SciSci's CPU does general computing with encodings from $\mathbb{Q}_2$, whose coefficients take values 0 or 1, and thus are bit strings compatible with Boolean logic gates.

Progress to Date

In six months, SciSci developed a virtual machine for the exact CPU. It has virtual processor registers and virtual arithmetic logic units with circuits composed of logic gates, which perform exact arithmetic. It accepts its own assembly language inputs and returns machine code outputs which, when sent to the compiler, give exact results. Addition, subtraction, multiplication, division, exponentiation, and roots are all shown in a recently released demo.