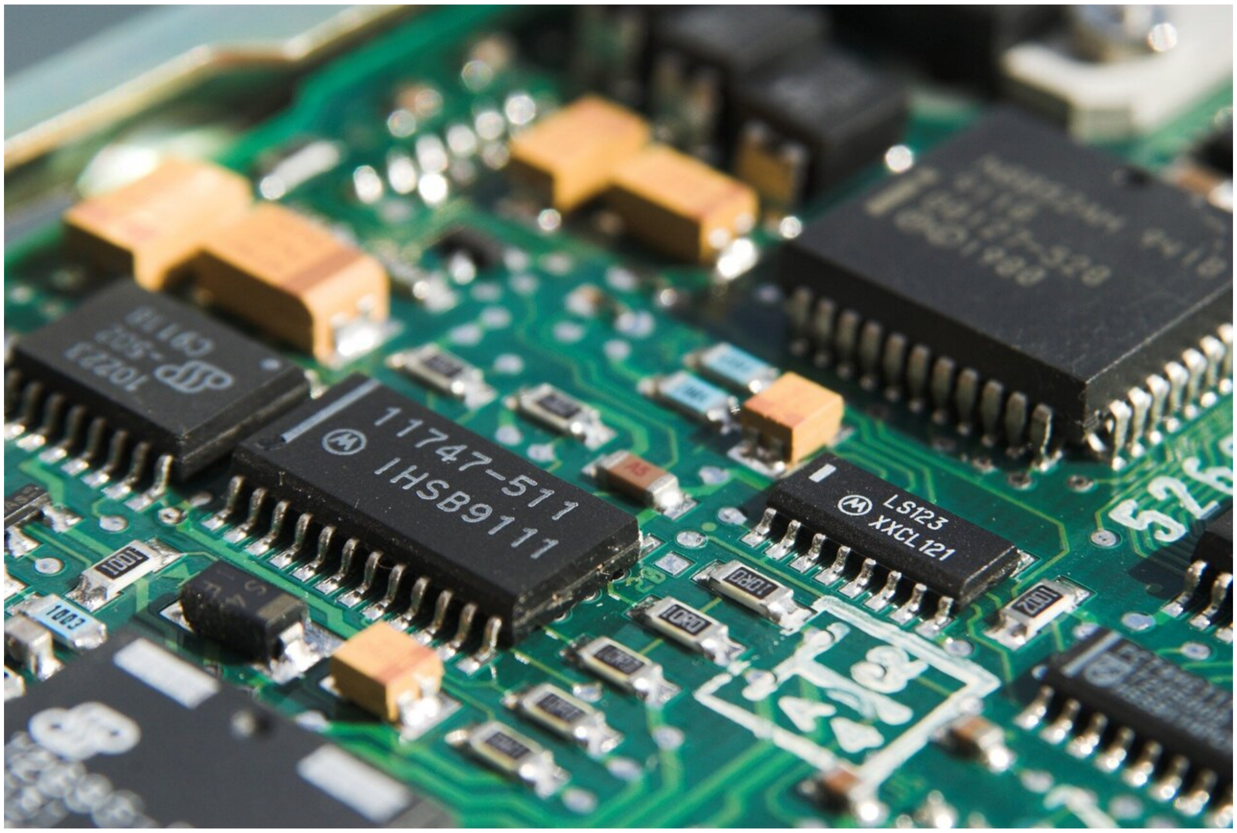
A new programming language for hardware accelerators

July 11 2022, by Rachel Gordon



Researchers created Exo, which helps performance engineers transform simple programs that specify what they want to compute into very complex programs that do the same thing as the specification, only much, much faster. Credit: Pixabay/CC0 Public Domain

Moore's Law needs a hug. The days of stuffing transistors on little silicon computer chips are numbered, and their life rafts—hardware accelerators—come with a price.

When programming an accelerator—a process where applications offload certain tasks to system hardware especially to accelerate that task—you have to build a whole new software support. Hardware accelerators can run certain tasks orders of magnitude faster than CPUs, but they cannot be used out of the box. Software needs to efficiently use accelerators' instructions to make it compatible with the entire application system. This translates to a lot of engineering work that then would have to be maintained for a new chip that you're compiling code to, with any programming language.

Now, scientists from MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) created a new programming language called "Exo" for writing high-performance code on hardware accelerators. Exo helps low-level performance engineers transform very simple programs that specify what they want to compute, into very complex programs that do the same thing as the specification, but much, much faster by using these special accelerator chips. Engineers, for example, can use Exo to turn a simple matrix multiplication into a more complex program, which runs orders of magnitude faster by using these special accelerators.

Unlike other programming languages and compilers, Exo is built around a concept called "Exocompilation." "Traditionally, a lot of research has focused on automating the optimization process for the specific hardware," says Yuka Ikarashi, a Ph.D. student in electrical engineering and computer science and CSAIL affiliate who is a lead author on a new paper about Exo. "This is great for most programmers, but for performance engineers, the compiler gets in the way as often as it helps. Because the compiler's optimizations are automatic, there's no good way to fix it when it does the wrong thing and gives you 45 percent efficiency instead of 90 percent."

With Exocompilation, the performance engineer is back in the driver's seat. Responsibility for choosing which optimizations to apply, when, and in what order is externalized from the compiler, back to the performance engineer. This way, they don't have to waste time fighting the compiler on the one hand, or doing everything manually on the other. At the same time, Exo takes responsibility for ensuring that all of these optimizations are correct. As a result, the performance engineer can spend their time improving performance, rather than debugging the complex, optimized code.

"Exo language is a compiler that's parameterized over the hardware it targets; the same compiler can adapt to many different hardware accelerators," says Adrian Sampson, assistant professor in the Department of Computer Science at Cornell University. "Instead of writing a bunch of messy C++ code to compile for a new accelerator, Exo gives you an abstract, uniform way to write down the 'shape' of the hardware you want to target. Then you can reuse the existing Exo compiler to adapt to that new description instead of writing something entirely new from scratch. The potential impact of work like this is enormous: If hardware innovators can stop worrying about the cost of developing new compilers for every new hardware idea, they can try out and ship more ideas. The industry could break its dependence on legacy hardware that succeeds only because of ecosystem lock-in and despite its inefficiency."

The highest-performance computer chips made today, such as Google's TPU, Apple's Neural Engine, or NVIDIA's Tensor Cores, power scientific computing and machine learning applications by accelerating something called "key sub-programs," kernels, or high-performance computing (HPC) subroutines.

Clunky jargon aside, the programs are essential. For example, something called Basic Linear Algebra Subroutines (BLAS) is a "library" or collection of such subroutines, which are dedicated to linear algebra computations, and enable many machine learning tasks like neural networks, weather forecasts, cloud computation, and drug discovery. (BLAS is so important that it won Jack Dongarra the Turing Award in 2021.) However, these new chips—which take hundreds of engineers to design—are only as good as these HPC software libraries allow.

Currently, though, this kind of performance optimization is still done by hand to ensure that every last cycle of computation on these chips gets used. HPC subroutines regularly run at 90 percent-plus of peak theoretical efficiency, and hardware engineers go to great lengths to add an extra five or 10 percent of speed to these theoretical peaks. So, if the software isn't aggressively optimized, all of that hard work gets wasted—which is exactly what Exo helps avoid.

Another key part of Exocompilation is that performance engineers can describe the new chips they want to optimize for, without having to modify the compiler. Traditionally, the definition of the hardware interface is maintained by the compiler developers, but with most of these new accelerator chips, the hardware interface is proprietary. Companies have to maintain their own copy (fork) of a whole traditional compiler, modified to support their particular chip. This requires hiring teams of compiler developers in addition to the performance engineers.

"In Exo, we instead externalize the definition of hardware-specific backends from the exocompiler. This gives us a better separation between Exo—which is an open-source project—and hardware-specific code—which is often proprietary. We've shown that we can use Exo to quickly write code that's as performant as Intel's hand-optimized Math Kernel Library. We're actively working with engineers and researchers at several companies," says Gilbert Bernstein, a postdoc at the University of California at Berkeley.

The future of Exo entails exploring a more productive scheduling meta-language, and expanding its semantics to support parallel programming models to apply it to even more accelerators, including GPUs.

Ikarashi and Bernstein wrote the paper alongside Alex Reinking and Hasan Genc, both Ph.D. students at UC Berkeley, and MIT Assistant Professor Jonathan Ragan-Kelley.

More information: Yuka Ikarashi et al, Exocompilation for productive programming of hardware accelerators, *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (2022). DOI: 10.1145/3519939.3523446

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology

Citation: A new programming language for hardware accelerators (2022, July 11) retrieved 5 May 2024 from https://techxplore.com/news/2022-07-language-hardware.html