RETROSPECTIVE: General-Purpose Code Acceleration with Limited-Precision Analog Computation

Renée St. Amant Amir Yazdanbakhsh[§] Jongse Park[º] Hadi Esmaeilzadeh*
Arjang Hassibi** Luis Ceze[†] Doug Burger[‡]

[§]Google DeepMind [º]KAIST *University of California San Diego
**Siomyx [†]University of Washington [‡]Microsoft

renee.st.amant@gmail.com ayazdan@google.com jongse@kaist.ac.kr hadi@ucsd.edu
arjang.hassibi@gmail.com luisceze@cs.washington.edu dburger@microsoft.com

When this paper was published in 2014, a large amount of research was focusing on specialization as a means to deliver energy efficiency. In this piece of work, we were searching for an alternative path to deliver significant efficiency gains while maintaining generality and applicability across domains. In order to do this, we relaxed some long-standing assumptions around exact computation and embraced approximation. A move towards approximation naturally warranted a revisit of analog computing, which presents an opportunity for energy efficiency gains of several orders of magnitude over digital computing when sufficient parallelization exists, though analog computing brings along many challenges (e.g. physical range limitations, non-idealities due to process variation, and noise), which have historically limited general applicability.

One key insight of this work was identifying a synergy with neural networks, which allowed for a more fixed-function, parallel design amenable to an analog implementation while maintaining generality, as neural networks can serve as universal function approximators. Ultimately, we took a neural approach to reconcile the application of analog circuits to general-purpose computing. Our paper proposed a limited-precision, analog neural accelerator (A-NPU), leveraging an important compile time technique [18] that transformed approximation-tolerant, general-purpose code sections to neural accelerator invocations. To improve accuracy given an analog implementation, we exposed certain analog hardware limitations to the compiler, which brought the opportunity to keep any analog-specific restrictions hidden from the programmer.

What strikes us most in reflecting on this piece of work is how the environment in which this paper was written was so completely different than today's environment. In nine years, so much has changed. In an ideal world, with the passage of time, we would hopefully gain some clarity on whether the risks we took against consensus thinking were worthwhile. We are fortunate in this case to have some, but not complete, validation in the presence of hindsight.

The interesting part of this work at the time was that it involved not one calculated risk but multiple risks taken at the same time that were contrary to the consensus approach. First, the paper was based heavily on the use of machine learning at a time when machine learning was not yet widely embraced by the architecture community. Next, it fell on us to make a case about the role of approximate computing and limited precision. And finally, we chose to include analog computing, which is in equal parts promising and challenging to successfully implement. As we look back on this paper, we can see that some of these risks were entirely appropriate to be taken, and, at least in the case of analog, some of the promise has yet to be fully realized.

I. Machine Learning

With respect to the inclusion of machine learning, the first drafts of the paper included much more ML content than what ultimately appeared in the published paper. An interesting, behind-the-scenes fact about this paper is that we did something that would be unheard of today to increase the paper's chance of being published - we removed much of the detailed machine learning content and the references to it that the work entailed. In reality, much of the difficult problem solving centered around making machine learning modifications in training and inference that were amenable to an analog implementation, for example, limiting connectivity in the neural architecture, overcoming limitations in implementing the neural activation functions, as well as developing quantization strategies for use during training. This research occurred before the widespread integration of efficient activation functions, before auto-ML tools existed, and before modern ML frameworks and libraries were available. Looking back, everything was much more tedious than it would be today, and it has been impressive to witness how much work has been done with machine learning in the last nine years.

While we built upon the potential to use neural networks for efficient general-purpose computing and took steps to enable this in the analog domain, today we could go as far as to say that the computing industry has now solidified a second

major class of general-purpose computing. While the first is precise, mostly deterministic code written in imperative languages to run on CPUs, the second is neural networks learning semantic hierarchies and patterns from structured and unstructured data. Nine years ago, it would have been premature to call this a second general class of computation, but not anymore. Regardless, the relevance of ML to the architecture community is now undeniable and remains a source of continuing innovation in translating compute cycles to utility.

II. Approximate Computing and Limited Precision

Another risk that we took involved having to make the case for approximate computing, including limited precision computation. We had a vision that approximate computing could someday be practically adopted in general-purpose applications. These applications include not only machine learning-based prediction applications but also other applications from many different domains (e.g., image processing, finance, robotics, etc.). We focused on approximate computing at the application level applied to general-purpose imperative code, which was, in some ways, the most extreme position to explore. We took that route because it offered the largest potential benefit for energy savings across the widest range of computing applications. At the time, the capabilities of limited-precision machine learning were largely unexplored, designs were over provisioned by default, and there was very little work looking at limited-precision approaches for regression tasks in particular (as opposed to classification tasks).

Today, approximation is an essential part of machine learning acceleration, and limited precision is one key effort. The use of approximation in cases where some level of imprecision can be tolerated is now a well-traveled approach to gain energy efficiency. And, in our opinion, the utility of approximation as a fruitful design path has also been validated.

III. Analog

The last significant risk that we took was the use of analog computing to maximize gains in energy efficiency. In the presence of sufficient parallelism and/or efficient analog storage, the move from digital computation to analog computation presents an opportunity for gains in energy efficiency on the order of 100x. Our work in this paper did not fully deliver on the available gains due to immaturity in analog storage options and the resulting overheads associated with digital-to-analog and analog-to-digital conversions, which, unsurprisingly, limited the gains that we were able to realize at the time by an order of magnitude. While we have seen investment and industry efforts in analog implementations of machine learning hardware, this component is the one that is most likely to be seen as still playing out and yet to be fully validated.

Although we took a calculated, neural approach to transcend the analog challenges associated with programmability, programmability remains an ongoing challenge for the successful incorporation of analog circuits into general-purpose computing. In some ways, we are in the same position today as we were nine years ago with analog. That is, despite its enormous potential, analog remains hard to pull off and perhaps the jury is still out on whether a return on investment in this area is imminent. While progress in this area is moving slower and all of the necessary pieces haven't yet fallen into place, we hope to see this thread eventually realize its full potential.

IV. Conclusion

Looking forward, we may find that the capabilities enabled by large-scale machine learning models allow a more aggressive, wholesale application of the techniques in this paper to general-purpose code at even larger granularities. Furthermore, efficiency continues to be a huge and growing imperative. Under our current ML trajectory, we may create large models exceeding the number of connections in the human brain, but we are nowhere near the energy efficiency of the human brain and won't be without a shift in direction - possibly towards analog. As such, the goalpost that we used nine years ago along with the call for questioning assumptions is still a guiding goalpost today when the primary concern is energy efficiency. We anticipate decades more of architecture+algorithmic work ahead. Some of that may move into the analog domain (where the neocortex already lives) or it may not due to the process gap and challenges with analog hardware.

While neural networks can serve as "universal function approximators," it's important to note that a "function" is not limited to a function call, but rather can refer to a higher-level problem to be solved. In the digital NPU work that introduced the idea of a compile-time transformation from general-purpose, imperative code to a neural accelerator invocation [18], as well as in our efforts undertaken while doing the research for this publication, we aimed to make this "function" as large as possible to maximize the available potential efficiency gains. Moving forward, we might see this further expand across the compute stack. As a visual reference, consider Yale Patt's classic compute stack image, which illustrates the stack layers between the topmost layer, i.e. the problem, and the bottommost layer, i.e. the electrons moving around to solve it. In the future, we may find that image being redrawn with ML compressing layers in a way that introduces more approximation and significantly improves energy efficiency (hopefully not at the expense of interpretability).

Thank you to the ISCA program committee members who selected this work as representative of how research in computer architecture has progressed over the last 25 years. And thank you to those researchers who continue to challenge the architecture community's long-standing assumptions and consensus approaches. We look forward to seeing how continued challenges to specific assumptions will shape the field in the next 25 years.