Retrospective: Neural Cache: Bit-Serial In-Cache Acceleration of Deep Neural Networks [1]

Charles Eckert, Xiaowei Wang, Jingcheng Wang, Arun Subramaniyan,
Ravi Iyer†, Dennis Sylvester, David Blaauw, Reetuparna Das
University of Michigan †Intel Corporation
{eckertch, xiaoweiw, jiwang, arunsub, dmcs, blaauw, reetudas}@umich.edu, ravishankar.iyer@intel.com

Traditional computer designs have always distinguished between the roles of storage and computation. While memories and caches are responsible for data storage, processors' logic units handle computation. But is this segregation indispensable? The human brain doesn't distinctly demarcate between the two functions. So, why should a processor? Our work raises this fundamental question regarding the role of caches.

A New Design Paradigm. Until today, caches have served only as an intermediate low-latency storage unit. Our work directly challenges this conventional design paradigm, and proposes to impose a dual responsibility on caches: store *and* compute data. By doing so, we turn them into massively parallel vector units, and drastically reduce on-chip data movement overhead.

Our work has its roots in the processing-in-memory (PIM) line of work [1]. PIMs move logic *near* main memory (DRAM), and thereby reduce the gap between memory and compute units. *Neural Cache* in contrast, morphs cache (SRAM) structures into compute units, keeping data *in-place*.

Developing the idea. The idea has its foundation in our prior work, Compute Cache [2]. With Compute Cache, we established the feasibility of logical operations on SRAM bit-lines and demonstrated a handful of use cases. However, after Compute Cache was published, critical questions remained unanswered: Can this technology support more than logical operations? What would be a killer application?

An insight then dawned on us - if we can execute logical operations in SRAM, we should also be able to perform complex arithmetic *using* in-memory logical operations. For a while, we were hindered by the complexity of bit propagation among bit-lines, which undermined the beauty of the proposition. In fact, we even attempted analog computing. However, without the use of expensive ADCs, this technology failed miserably due to variation effects.

The idea of using bit-serial computation was our breakthrough while scourging through older digital logic literature. Our work took a unique architectural approach to in-memory computing by employing compact bit-serial logic over transposed data. It processes one bit at a time, eliminating the need for expensive ADCs and enabling higher precision. Since only a binary value needs to be sensed every cycle, existing SRAM array sense amps can be utilized. Bit-serial computing over bit-lines obviates the need for communication across bit-lines for carry propagation, keeping the logic compact. Other approaches towards in-memory computing [3] utilize analog computing requiring expensive ADCs and restricted to limited precision.

At this stage, we had developed a microcode library of In-SRAM arithmetic operations along with a comprehensive circuit design for array peripherals and a transposed memory unit. Concurrently, we delved into the geometry of modern caches, gaining deep insights into their layout, array peripherals, sense-amp multiplexing, and data array organization. Suffice to say, the layout of the last level cache in modern CPUs significantly deviates from textbook set-associative caches or even CACTI. With our breakthrough, we could now potentially repurpose a 35 MB LLC in the server-class Intel Xeon E5-2697 v3 to support 1,146,880 bit-serial ALU slots. This computational capability greatly exceeded the aggregate SIMD width in Xeon (448 32-bit slots), or even the then-latest GPU (Nvidia Titan Xp with 3840 32-bit slots).

So, what about the killer application? It was 2017 when DNNs were gaining significant popularity, and during a discussion at the ISCA PC meeting between two co-authors of our paper, the question arose: Could our technology accelerate DNNs? This became a fun challenge for our team to undertake. Subsequently, the ISCA 2017 tutorial by Joel Emer and Vivienne Sze was a wonderful avenue to understand the key computational primitives of CNNs.

Upon deeper introspection, it became apparent that our technology was an excellent fit for DNNs. Why so? Although our architecture exhibited high latency for individual operations, it could execute millions of MACs in parallel, which aligns well with the needs of unrolled CNNs, as they require the parallel execution of millions of MACs. What ensued was months of intensive research to learn CNNs, and develop a massively data-parallel computation mapping strategy onto the cache architecture.

[1]Originally appeared in IEEE International Symposium on Computer Architecture (ISCA), Los Angeles, CA, 2018. The full paper is available at https://dl.acm.org/citation.cfm?id=3276577

What we learned and what followed. In our paper, we demonstrated that *Neural Cache* can be employed to accelerate memory and compute-intensive programs such as neural networks. Performing arithmetic operations in-place in caches is a game changer, as it can transform CPU caches into accelerators for data and graph analytics. Our subsequent work, Duality Cache [4], designed support for floating point operations, developed a SIMT microarchitecture, and built a CUDA compiler for programmability. This allowed us to run *arbitrary* data-parallel programs entirely on cache while providing an order-of-magnitude efficiency gain.

Neural Cache's significant latency and energy improvements on DNNs inspired many follow-up research works citing our work, all of which pursued relevant in-memory approaches to design DNN acceleration solutions. However, *Neural Cache* is only one instance of the compute cache architecture paradigm. Our recent work [5] demonstrated that caches can be repurposed to accelerate automata processing, achieving speeds 15 times faster than Micron's DRAM-based Automata Processor, and 320 times faster than a GPU. Our follow-up work, GenCache [6], displayed similarly powerful results for Genomics. In addition to caches, our group demonstrated that In-SRAM computing is an exciting technology applicable to block RAMs in FPGA [7] and as an independent ASIC accelerator for LSTMs [8].

We pioneered In-SRAM computing in the research community six years ago. The topic continues to attract significant attention in the circuits/architecture community and has expanded to include other emerging on-chip memories. A distinctive feature of Neural Cache is its use of bit-serial computing. Numerous subsequent works continue to demonstrate that bit-serial computing can enhance in-memory architectures (e.g., [9], [10], [11], [12]).

Looking forward. Given that caches are present in nearly all modern processors, we envision *Neural Cache* as a revolutionary technology that can significantly augment commodity processors with large data-parallel accelerators at virtually no extra cost. This advancement enables CPU vendors (such as Intel, AMD, IBM, etc.) to maintain their delivery of high-performance general-purpose processing while supplementing it with a co-processor-like capability designed to exploit massive data parallelism. This type of processor design is particularly appealing for challenging-to-accelerate applications that frequently oscillate between sequential and data-parallel computation. Moreover, the recent trend towards increasing cache sizes renders the proposed technology even more compelling [13]. For instance, recent processors boast a total cache size of several 100 MBs! [14]. To conclude, nearly three-fourths of a server class processor die area today is devoted to caches. Even accelerators use large caches. Why would one *not* want to turn them into compute units?

References

[1] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "A case for intelligent ram," *Micro, IEEE*, 1997.

[2] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute caches," in *HPCA-23*, 2017.

[3] P. Srivastava, M. Kang, S. K. Gonugondla, S. Lim, J. Choi, V. Adve, N. S. Kim, and N. Shanbhag, "Promise: an end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms," in *ISCA-45*, 2018.

[4] D. Fujiki, S. Mahlke, and R. Das, "Duality cache for data parallel acceleration," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: ACM, 2019, pp. 397–410. [Online]. Available: http://doi.acm.org/10.1145/3307650.3322257

[5] A. Subramaniyan, J. Wang, E. Balasubramanian, D. Blaauw, D. Sylvester, and R. Das, "Cache automaton," in *MICRO-50*, 2017.

[6] A. Nag, C. Ramachandra, R. Balasubramonian, R. Stutsman, E. Giacomin, H. Kambalasubramanyam, and P.-E. Gaillardon, "Gencache: Leveraging in-cache operators for efficient sequence alignment," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019.

[7] X. Wang, V. Goyal, J. Yu, V. Bertacco, A. Boutros, E. Nurvitadhi, C. Augustine, R. R. Iyer, and R. Das, "Compute-capable block rams for efficient deep learning acceleration on fpgas," in *29th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2021, Orlando, FL, USA, May 9-12, 2021*, 2021.

[8] C. Eckert, A. Subramaniyan, X. Wang, C. Augustine, R. Iyer, and R. Das, "Eidetic: An in-memory matrix multiplication accelerator for neural networks," *IEEE Trans. Computers*, vol. 72, no. 6, 2023.

[9] M. F. Ali, A. Jaiswal, and K. Roy, "In-memory low-cost bit-serial addition using commodity dram technology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 1, pp. 155–165, 2019.

[10] S. Angizi, Z. He, A. Awad, and D. Fan, "Mrima: An mram-based in-memory accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1123–1136, 2019.

[11] N. Hajinazar, G. F. Oliveira, S. Gregorio, J. D. Ferreira, N. M. Ghiasi, M. Patel, M. Alser, S. Ghose, J. Gómez-Luna, and O. Mutlu, "Simdram: a framework for bit-serial simd processing using dram," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 329–345.

[12] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable sram-based digital bit-serial compute-in-memory macro for processing neural networks," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, 2021.

[13] R. Iyer, V. De, R. Illikkal, D. Koufaty, B. Chitlur, A. Herdrich, M. Khellah, F. Hamzaoglu, and E. Karl, "Advances in microprocessor cache architectures over the last 25 years," *IEEE Micro*, vol. 41, no. 6, pp. 78–88, 2021.

[14] "Amd epyc 7003 processors (data sheet)," "https://www.amd.com/system/files/documents/amd-epyc-7003-series-datasheet.pdf".