RETROSPECTIVE: Smart Memories: a modular reconfigurable architecture

Ken Mai[1], Tim Paaske[2], Nuwan Jayasena[3], Ron Ho[4], William J. Dally[5,6], and Mark Horowitz[5]

[1]Carnegie Mellon University, [2]Apple, [3]AMD, [4]Meta, [5]Stanford University, [6]NVIDIA

I. Introduction

This paper in ISCA 2000 sought to address what has become the fundamental conflict in modern VLSI design: for energy efficiency, one must tailor the computation dataflow to match the application, yet at the same time to be cost effective, the part must address a large market to amortize the system design NRE costs. Power – and more importantly power efficiency – were growing concerns and were predicted to become first-tier issues in upcoming designs. Over the past two decades, this conflict has only gotten worse as technology continued to shrink transistor cost, allowing design complexity to grow, but power per device was scaling slowly since Dennard's scaling had ended. Thus, in modern chips, power is the number one design concern, data movement dominates power dissipation, and wire delays have grown rapidly relative to logic gates.

To provide optimized dataflows and generality, Smart Memories proposed what would now be called a CGRA (coarse grain reconfigurable array) with two unique features:

- The ability to program/tailor the memory blocks as well as the processors
- Intermixing DRAM tiles and compute tiles on the same die

These capabilities resulted in a reconfigurable architecture that could be tailored to the application after manufacturing. The design eschewed the overheads typically associated with reconfiguration by inserting data steering logic at repeater breakpoints where gates were needed anyway for optimal interconnect performance.

The key to programmable memories was that the vast majority of specialized memory systems used common SRAM sub-arrays arranged in different ways. As such, we showed how to insert reconfigurability/customization at sub-array boundaries, avoiding disruptive changes to the core SRAMs and, once again, minimizing the overheads customarily associated with reconfigurability. This programmable memory was built successfully a few years later [8][9]. Finally, we implemented configurability in the processor datapaths to enable computation to be configured to meet the needs of the target application.

Work on the Smart Memory approach continued after this initial ISCA paper, leading to the development of the Smart Memories chip multiprocessor (SMASH), and finally to a full system implementation [1]. While the original paper was very VLSI centric, exploring how to add configuration into the basic building blocks, continuing this approach for the whole system design was problematic for a small university design team. Development of a custom processor core and ISA required development of a complete set of software tools and infrastructure, and the configurable memory required creating a new memory compiler. In addition, both new blocks would require creating extensive validation frameworks capable of handling the multitude of possible configurations. Instead, we chose to focus on the reconfigurable memory system architecture and used Tensilica processor cores, adding the needed functionality with custom instructions using the TIE language. This allowed us to add VLIW instruction formats and the mechanism to recover processor core state from mis-speculation in transactional memory mode [2]. High-quality software tools and infrastructure provided by Tensilica allowed us to compile and run verification tests and benchmark applications and to compare different modes, implemented in Smart Memories architecture, e.g., cache coherent and streaming modes [3].

For the reconfigurable memory system design the team leveraged the existing memory compiler from the foundry for the bulk of the SRAM data storage and synthesized the bespoke portions of the memory architecture using standard cells. Memory Mat metadata was stored in standard cell flip-flops, rather than the custom SRAM cells used in the initial reconfigurable memory testchip [8][9]. Although this increased the overhead significantly, it drastically decreased the needed engineering effort for the reconfigurable memory block. The resulting design still retained the original ideas of creating a programmable memory system, however the goals of this memory system changed. During the 2000's, interest in transactional memory was growing, so demonstrating our programmable memories could support it became one of our goals. As validating the resulting programmable controller posed a complex difficult chalenge, we chose to reduce the problem to validating that the protocols implemented on the machine worked, which was still complex, but tractable. This led to a new validation approach, called a relaxed scoreboard [4], which enabled this validation. The resulting system was one of the first to support transactional memory.

II. What the paper got right

The paper correctly predicted the rising importance of energy efficiency, which is now critical in all systems, from small IoT embedded systems to powerful mobile platforms

to big-iron datacenters. It also correctly foresaw the rise of spatially configurable computing (CGRAs) to more efficiently match an algorithm's dataflow, distributing compute and memory around the die to enable one to store data closer to where it is needed. Finally it was an earlier entry in the compute in/near memory paradigm, which continues to be of interest.

While the specific programmable memories that were proposed in this paper were never built into a full system, the need for creating some programmability in the memory system has been validated in many systems. For example, a common pool of memory that can be used as cache or scratchpad based on application needs is now a mainstream feature in GPUs [10]. Programmable memories are also used in most streaming accelerators. These machines generally use memories which push a data stream into a computing fabric and have another memory store the resulting output. In these systems, the addressing, flow control, and storage management is often "built in" to the memory, creating programmable memory units [11].

Notably today, machine learning accelerators, which are based on varying size Tensors, are a prime example of memories that are programmable and deeply embedded close to the computation. Since these memories are spatially arranged between arrays of compute units/ALUs, the memory structure itself is (re)programmed to serve as a virtual FIFO of the exact tensor dimensions [5-7]. The Neural Network graph compiler does the buffer properties assignment at the beginning of the computation. However, unlike Smart Memories, programmability here is typically limited to access patterns, rather than the consistency or coherence model.

III. What the paper missed

While the paper correctly pointed out the conflict between specialization and the need for a large market to amortize the high design costs, it failed to recognize an alternate approach to resolve this conflict by dramatically changing the way we design. If one could create a customized design for small design costs, the lack of generality would be less of a problem. This is important since all reconfigurable designs seem to have significant overheads. Fundamentally, the overhead of the reconfigurable design comes from either over-engineering it for yet to be known use cases (taking too long) or from mispredicting how use cases will evolve (missing a critical need). So, another approach to this conflict was to create tools which allowed designers to customize their designs from a more general chip generator. This approach was best exemplified by the Berkeley work on Chisel [12] and ChipYard. In fact, the Smart Memory design team took this approach to create the silicon we eventually taped-out [1].

Acknowledgements

The authors would like to acknowledge contributions to this retrospective by Zain Asgar, Amin Firoozshahian, Ofer Shacham, Alex Solomatnikov, and Megan Wachs.

References

[1] A. Firoozshahian, A. Solomatnikov, O. Shacham, Z. Asgar, S. Richardson, C. Kozyrakis, M. Horowitz. A Memory System Design Framework: Creating Smart Memories. 36th International Symposium on Computer Architecture (ISCA), p. 406-417, 2009.

[2] A. Solomatnikov, A. Firoozshahian, O. Shacham, Z. Asgar, M. Wachs, W. Qadeer, S. Richardson, M. Horowitz. Using a Configurable Processor Generator for Computer Architecture Prototyping. 42nd International Symposium on Microarchitecture, p. 358-369, 2009.

[3] J. Leverich, H. Arakida, A. Solomatnikov, A. Firoozshahian, M. Horowitz, C. Kozyrakis. Comparing Memory Systems for Chip Multiprocessors. 34th International Symposium on Computer Architecture (ISCA), p. 358-368, 2007.

[4] O. Shacham, M. Wachs, A. Solomatnikov, A. Firoozshahian, S. Richardson, M. Horowitz. Verification of Chip Multiprocessor Memory Systems Using A Relaxed Scoreboard. 41st International Symposium on Microarchitecture, p. 294-305, 2008.

[5] Norman P. Jouppi, et al. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17). Association for Computing Machinery, New York, NY, USA, 1–12.

[6] N. P. Jouppi et al., "Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product," 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 2021, pp. 1-14, doi: 10.1109/ISCA52012.2021.00010.

[7] S. Lie, "Cerebras Architecture Deep Dive: First Look Inside the Hardware/Software Co-Design for Deep Learning," in IEEE Micro, vol. 43, no. 3, pp. 18-30, May-June 2023.

[8] K. Mai et al., "Architecture and circuit techniques for a 1.1-GHz 16-kb reconfigurable memory in 0.18-/spl mu/m CMOS," in IEEE Journal of Solid-State Circuits, vol. 40, no. 1, pp. 261-275, Jan. 2005.

[9] K. Mai et al., "Architecture and circuit techniques for a reconfigurable memory block," 2004 IEEE International Solid-State Circuits Conference, San Francisco, CA, USA, 2004, pp. 500-542.

[10] "NVIDIA Tesla V100 GPU Architecture," https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

[11] Qiaoyi Liu, Jeff Setter, Dillon Huff, Maxwell Strange, Kathleen Feng, Mark Horowitz, Priyanka Raina, and Fredrik Kjolstad. 2023. Unified Buffer: Compiling Image Processing and Machine Learning Applications to Push-Memory Accelerators. ACM Trans. Archit. Code Optim. 20, 2, Article 26 (June 2023), 26 pages.

[12] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. 2012. Chisel: constructing hardware in a Scala embedded language. In Proceedings of the 49th Annual Design Automation Conference (DAC '12). Association for Computing Machinery, New York, NY, USA, 1216–1225.