Tileable Monolithic ReRAM Memory Design

Meenatchi Jagasivamani*, Candace Walden*, Devesh Singh*, Luyi Kang*,
Mehdi Asnaashari†, Sylvain Dubois†, Bruce Jacob*, and Donald Yeung*

*Department of Electrical & Computer Engineering, University of Maryland, College Park, MD, 20742.
†Crossbar Incorporated, Santa Clara, CA, 95054.

Non-volatile memory, such as resistive RAM (ReRAM), is compatible with standard CMOS logic processes, allowing a sizable main memory system to be integrated into a CPU's die. ReRAM bitcells are fabricated within *crosspoint sub-arrays* that leave the bulk of transistors underneath the sub-arrays vacant. This permits placing the memory system over other logic. We propose a tileable, centralized ReRAM design over a large last level cache. This design takes advantage of ReRAMs unique characteristics while still providing flexibility to designers.

Keywords: Crosspoint architectures, ReRAM and on-die main memory systems.

Recently there has been a lot of focus on emerging non-volatile memory in the memory hierarchy and 3D integration. This has led to the development of non-volatile memories that allow for 3D stacking of the memory cells to improve density. Examples include Intel's 3D XPoint [1] and Crossbar's 3D ReRAM [2]. In these 3D memory architectures, called "crosspoint architectures", the memory bitcells are sandwiched in between metal wires and individual bitcells are isolated by per-cell "selector devices" rather than access transistors. The use of selector devices enables extremely small bitcells that can be stacked vertically across multiple metal layers. It also means the transistors underneath these sub-arrays are free for implementing unrelated circuits. Some logic is still needed for access circuitry, but the bulk of the transistors are unused.

The benefits of these memory technologies include higher densities, thus capacities, than DRAM; lower power, as refresh is no longer required; and CMOS compatibility. Whereas DRAM requires special VLSI processes tuned for implementing DRAM's memory cells, 3D crosspoint sub-arrays can be fabricated today in commercial CMOS fabs at the same technology node as the underlying logic [2]. This implies the crosspoint memory can be fabricated over the CPU, occupying the top-level metal layers of the CPU's die, creating a monolithically integrated CPU–main memory chip. Meanwhile, the CPU's logic can be implemented in the die's logic transistors, minus those needed for the memory access circuits. Putting the CPU and its memory system on the same die will significantly reduce the energy to access memory, improving power efficiency. It will also allow for an extremely wide connection between the cores and the memory system which will benefit highly parallel architectures, such as tiled CPUs. This will provide much higher throughput and performance for data-intensive computations.

The drawback, though, is that these memories have higher latency than DRAM and suffer wearout. A popular approach proposed by many researchers is to retain a small amount of DRAM to buffer frequently accessed data [3], [4]. Additionally the entire main memory and CPU must fit on a single die. Integrating the two in a 2D planar fashion leads to the available area becoming a limiting factor necessitating 3D integration of the memory and CPU logic. This requires that the memory access circuitry be accounted for in CPU layout, possibly creating higher design complexity.

Previous work has looked at integrating general CPU logic under crosspoint arrays [5]. They found that placing and routing the CPU required an additional 18% on top of the area requirements of the memory access circuits–with only half the die occupied by ReRAM arrays. The irregular structure of the cores is not a good fit for this kind of integration–regular structures like cache do much better. The design we propose is a modular implementation of ReRAM over a last level cache. This could be tiled across the center of the die, with the edges containing CPU cores, DRAM controllers or even accelerators depending on the requirements. An example floorplan of such a system is shown in Fig. 1.

The centrally located ReRAM block in Fig. 1 is designed to act as a single embedded memory IP with a separate internal NoC. In addition to the NoC router circuits, the area underneath the ReRAM array could be used for SRAM arrays that can act as the last level cache. Designers only need to consider the number of memory banks, the size of the interconnect and its topology. Once an optimal configuration is selected, we can generate the overall ReRAM embedded block design and external interface block. This block can be used to generate a floorplan layout and provide area estimates to identify placement of individual components to achieve such a system.

For the repeating block, the ReRAM arrays will be laid in 4 groups of 4, creating a tiled cross as shown in Fig. 3. The memory controller, bank controller, NoC router are placed in the free area beneath the central ReRAM arrays; SRAM cache can be placed beneath the surrounding arrays.

The ReRAM memory controller coordinates accesses to n banks, where n is selected as a tradeoff between fine-grain memory accesses and area overhead. For our example design, we have selected n to be 8 which provides 64-bit data accesses. If wider accesses are desired, the memory controller will be capable of supporting streaming mode which perform a burst of accesses to 8 adjacent banks, matching DRAM granularity and improving streaming bandwidth. In addition to circuitry directing the banks, the memory controller would contain an incoming request queue of 32B to support eight requests of 32-bit commands, control logic to reorder pending requests, and a data buffer of 256B to store read and write data. Fig. 2 shows the design for the ReRAM memory controller.

Each bank will contain a bank controller. The bank controller logic has three main functions, as shown in Fig. 4: an incoming request queue, a circuit to initiate the read and write kickoff signals to all 16 arrays, and a data buffer to store the read and write data. We model a 32B register file for the incoming request to support eight 32-bit command requests. A 64B register file is used to model the data-buffer to store pending read and write data.

For the NoC, we propose a torus topology with a 32B interconnect width. We allocate 16B for the ReRAM main memory and 16B for the SRAM cache in order to keep the two memory systems separate and to allow for different priorities and address schemes to be implemented between them. Such an interconnect gives nearly maximum performance for fine-grain pointer chasing kernels and 90% of performance for stream-type computations under full load.

Based on the previous layout study, which placed a small core under 4 MB 2-level stacks of ReRAM at the 45 nm technology node, we estimate a standard cell efficiency of 60%. The processor logic required $240,000\mu\mathrm{m}^2$ of the total $400,000\mu\mathrm{m}^2$ ($625\mu\mathrm{m} \times 625\mu\mathrm{m}$) area shown in Fig. 3.

To estimate the areas for the bank and memory controller, we synthesized a representative Verilog file to model the functions and used it to estimate the automatic place and route (APR) area. We extrapolate this by using the area reported from the small core layout study where the standard cell area was $22,088\mu\mathrm{m}^2$, and the APR area was $30,373\mu\mathrm{m}^2$. The synthesized area for the bank controller was $4,162\mu\mathrm{m}^2$, which translates to $5,723\mu\mathrm{m}^2$ after the APR step. As a square block, this circuit could be expected to take up $76\mu\mathrm{m} \times 76\mu\mathrm{m}$ of area underneath the ReRAM array. When synthesized, memory controller reported a total area of $16,147\mu\mathrm{m}^2$, which we extrapolate to be $22,204\mu\mathrm{m}^2$ after the APR step. Since this logic block will be shared among 8 banks, this circuit could be expected to use a square footprint of $18\mu\mathrm{m} \times 18\mu\mathrm{m}$ for each bank.

The bank, memory, and NoC controllers will be placed in a central area of the ReRAM mat, which has an area of $343\mu\mathrm{m} \times 343\mu\mathrm{m}$. Fig. 5 shows the relative sizes and placement of the bank controller (B), memory controller (M) being which is shared among 8 banks on the ReRAM mat. The remaining area in the block can be used for the NoC router.

SRAM arrays will surround the central area with size of $125\mu\mathrm{m} \times 125\mu\mathrm{m}$. These SRAM arrays can be used as the last-level cache on the chip and can operate independently from the main-memory ReRAM control logic. In one bank, we can fit 3 of these SRAM arrays. Assuming 75% array efficiency and using the academic OpenRAM bitcell which has a size of $1.344\mu\mathrm{m} \times 0.707\mu\mathrm{m}$, the total SRAM storage per array is 4.1kB. A commercial version of the SRAM bitcell could foreseeably be drawn 2.5x smaller, and thus achieve 10kB of SRAM capacity per array.

Finally, we consider the routing channel to connect the main-memory and the independently operating SRAM cache memories to the NoC router endpoints. The NoC interconnects can be drawn in metal-7 and metal-8 which are available to be used in the regions between the ReRAM arrays and over the SRAM arrays. This is illustrated in Fig. 6 below. The horizontal tracks are metal-7 and the vertical tracks are in metal-8 and these would provide a global interconnect channel to the ReRAM and SRAM arrays. The available spacing for this in the floorplan above is 125um wide as thats the spacing between the ReRAM arrays. To achieve the target 32B channels, a metal pitch is required to be .5um in the metal-7 and metal-8 which should be achievable in this technology.

[1] Intel, "Intel Optane Technology," http://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html, 2017.

[2] Crossbar, "ReRAM Memory, Crossbar," https://www.crossbar-inc.com/assets/resources/white-papers/Crossbar-ReRAM-Technology.pdf, 2017.

[3] S. R. Dulloor, A. Roy, Z. Zhao, N. Sundaram, N. Satish, R. Sankaran, J. Jackson, and K. Schwan, "Data Tiering in Heterogeneous Memory Systems," in *Proc. 11th European Conf. on Computer Systems*, 2016.

[4] L. Ramos, E. Gorbatov, and R. Bianchini, "Page Placement in Hybrid Memory Systems," in *Proc. 2011 International Conf. on Supercomputing*, 2011.

[5] M. Jagasivamani, C. Walden, D. Singh, L. Kang, S. Li, M. Asnaashari, S. Dubois, D. Yeung, and B. Jacob, "Design for reram-based main-memory architectures," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 342350.

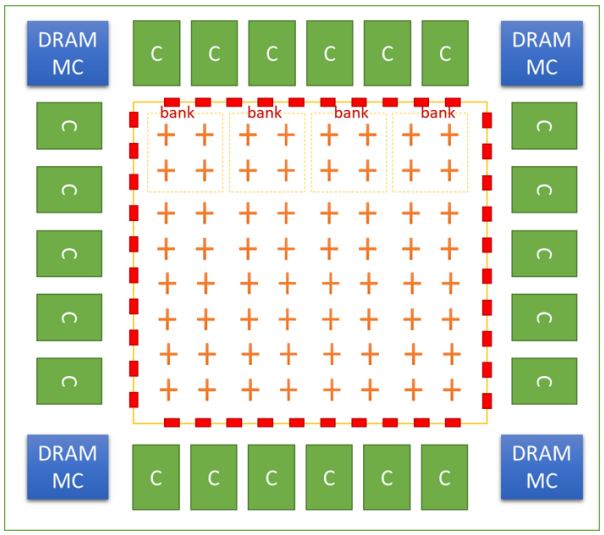

Fig. 1. Hybrid ReRAM-DRAM System Floorplan

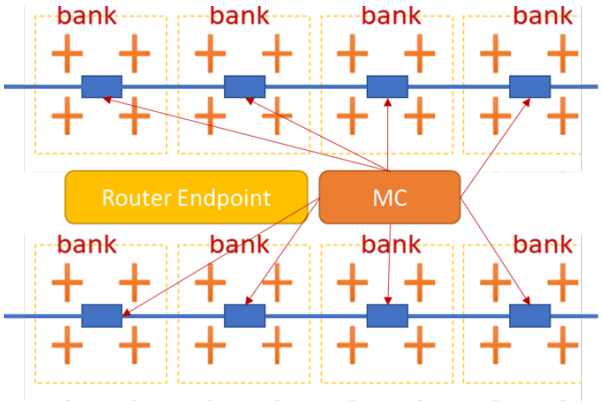

Fig. 2. ReRAM Memory Controller Design

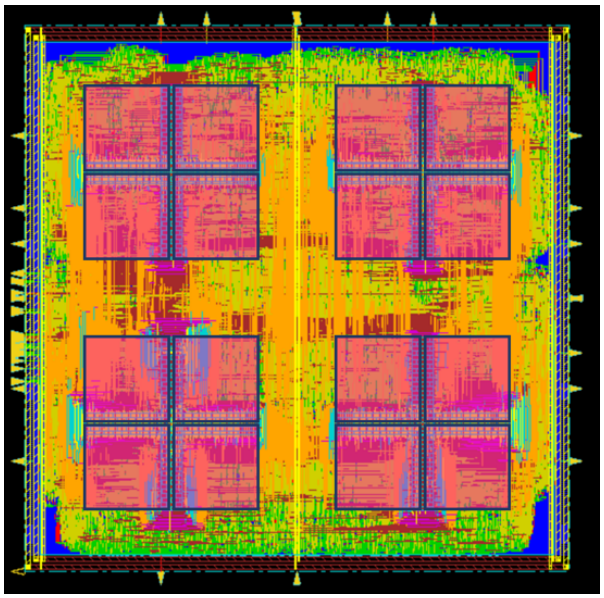

Fig. 3. Memory Footprint for Central ReRAM Design

Fig. 4. Memory Footprint for Central ReRAM Design

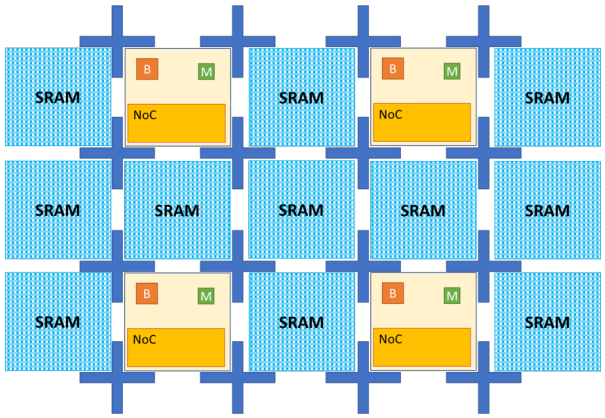

Fig. 5. Memory Footprint for Central ReRAM Design

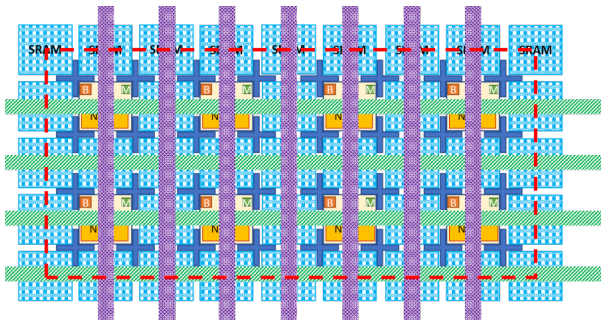

Fig. 6. Memory Footprint for Central ReRAM Design

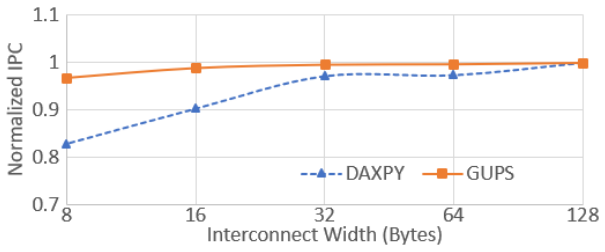

Fig. 7. Affect of interconnect width on performance.