Design for ReRAM-based Main-Memory Architectures

Meenatchi Jagasivamani
Candace Walden
Devesh Singh
Luyi Kang
Shang Li
Dept. of Electrical Computer Engineering
University of Maryland
College Park, MD

Mehdi Asnaashari
Sylvain Dubois
Crossbar, Inc.
Santa Clara, CA

Donald Yeung
Bruce Jacob
Dept. of Electrical Computer Engineering
University of Maryland
College Park, MD

ABSTRACT

With the anticipated scaling issues of DRAM memory technology and the increased need for higher density and bandwidth, several alternative memory technologies are being explored for the main memory system. One promising candidate is a variation of Resistive Random-Access Memory (ReRAM) which implements the memory bit-cells on Back-End-of-Line (BEOL) layers. This allows for fabrication of the processor logic and ReRAM main-memory to be implemented on the same chip. As the memory cells can be stacked vertically, the density of this memory also scales to 1-4F^2. This tight integration allows for a high amount of parallelism between the processor and memory systems and delivers low access granularity without sacrificing density or bandwidth.

In this paper, we explore physical integration of a processor with a ReRAM-based main-memory system using the bitcell technology developed by Crossbar, Inc. We present Crossbar's ReRAM technology characteristics, the methodology and assumptions used for our digital implementation, and summarize the results obtained for different array configurations. Our results indicate that, in addition to the overhead for the ReRAM access circuits, the overall integrated area increases by 11% to 19%, based on the configuration at the 45nm process node. Results from architectural simulation comparing DRAM with ReRAM based architecture are presented.

MEMSYS '19, September 30-October 3, 2019, Washington, DC, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-7206-0/19/09…$15.00
https://doi.org/10.1145/3357526.3357561

CCS CONCEPTS

• **Hardware** → *Non-volatile memory*; • **Computer systems organization** → **Embedded systems**.

KEYWORDS

ReRAM, main-memory, emerging technology, digital implementation, layout

ACM Reference Format:
Meenatchi Jagasivamani, Candace Walden, Devesh Singh, Luyi Kang, Shang Li, Mehdi Asnaashari, Sylvain Dubois, Donald Yeung, and Bruce Jacob. 2019. Design for ReRAM-based Main-Memory Architectures. In *Proceedings of the International Symposium on Memory Systems (MEMSYS '19), September 30-October 3, 2019, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357526.3357561

1 INTRODUCTION

The memory bus is a major limiting factor to the overall system performance. Current CPU performance is typically 3-4x faster than the rate delivered by overall memory system[7]. There are several mitigation strategies that are currently employed to address this problem, at the hardware (memory hierarchy), software (prefetch, managing access patterns), and at the system level (multiple memory controllers). However, we are fundamentally limited by the number of wires that connect the processor and the memory chip. This creates a bottleneck that backs up the flow of data between the two components, as shown in Figure 1. Therefore, even though the memory latency for a system is sufficiently fast, the overall perceived latency time might be slow. We see that this bandwidth wall stems from the limited number of memory access points that exist in current systems.

Several recent technologies have enabled a key innovation in computer architecture to replace traditional DRAM-based main-memory system with emerging non-volatile memory technologies and to help address the memory bandwidth

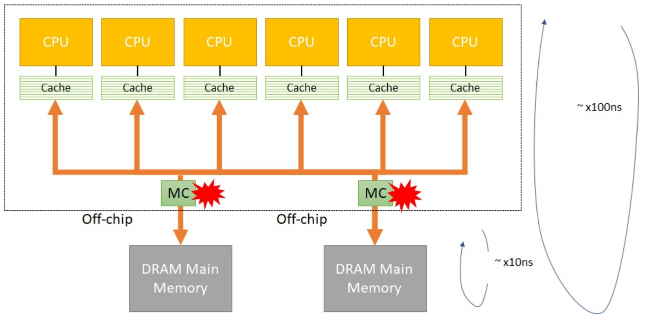


Figure 1: Limited Connections attributed to Memory Bandwidth Wall

problem. Our proposed solution involves integrating ReRAM as the main-memory for a CPU on the same chip. Note that this is different from 3D stacked-die types of approaches that make use of physical integration of discrete dies. Our solution involves the ReRAM memory bitcells residing in metal layers which are fabricated on the same die, above the processor logic. Figure 2 illustrates this difference by comparing a conventional architecture involving a separate CPU die (Figure 2a), versus that of a Monolithic Computer[1] where the CPU logic and the ReRAM main-memory reside on the same chip (Figure 2b). This technology has been demonstrated and fabricated at 40nm [2].

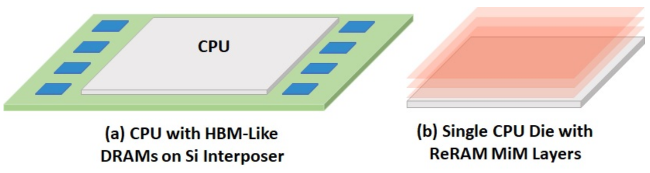

CPU
(a) CPU with HBM-Like DRAMs on Si Interposer
(b) Single CPU Die with ReRAM MiM Layers

Figure 2: Comparison of (a) Conventional with a (b) Monolithic Architecture

The rest of the paper is organized as follows. Section 2 summarizes the Crossbar ReRAM technology and the physical design integration constraints we used for our area study. Section 3 presents our methodology for the digital implementation, the results of an integrated ReRAM-processor design and of a SRAM-ReRAM design. Section 4 presents architectural study using SST to compare a DRAM-based system against a ReRAM-based architecture. Finally, section 5 summarizes the paper in conclusion.

2 CROSSBAR RERAM TECHNOLOGY

In this section we present a brief overview of Crossbar ReRAM characteristics, integration constraints relevant to this study, and the CAD tool methodology used.

2.1 ReRAM Physical Characteristics

The Crossbar bitcell couples the resistive switching medium with a FAST selector device with a high RÂňon/Roff ratio and integrates with standard CMOS processes. The bitcell relies on the creation of microscopic conductive filaments in the switching medium through ion migration for the resistive element [3, 4]. Figure 3(a) shows the cross section of the 1S1R (1 selector per 1 Resistive element) bitcell used. A proprietary FAST selector device enables high selectivity and fast access times. A voltage above a threshold (> VTH) is required to select the cell to perform a read or write operation. For program, a much higher voltage (>VPRG) is applied to enable the formation or resetting of the conductive filaments. Figure 3(b) shows the bias scheme of the crossbar memory array for selection. All wordlines and bitlines are held at V/2, while the selected cell's wordline and bitline are biased to have a difference of V across it. The high selectivity ($>10^6$-10^{10}) ensures minimal sneak path current on unselected cells on the same bitline, which have a potential of V/2 across their cells.

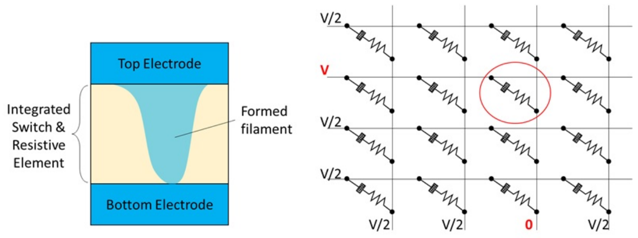


Figure 3: (a) 1S1R ReRAM bitcell cross-section (b) Crossbar array bias scheme, with a selected cell

Table 1 summarizes the key performance metrics of Crossbar's ReRAM array. During read, the voltages are expected to operate in the nominal range for the technology, while write voltages are expected to be pumped to a higher voltage levels. As noted in the table, the bandwidth per array is 4-8 bits. Therefore, to provide sufficient bandwidth to a single core, we envision several arrays that are distributed across the full-chip and are accessed in ganged mode, in a Distributed Shared Memory like architecture.

2.2 ReRAM Integration with Logic

The ReRAM memory developed by Crossbar is CMOS compatible and back end of line (BEOL) stackable. As mentioned in Section 1, this type of ReRAM is organized so that the bitcells are stacked on higher metal layers. The peripheral support circuitry to perform the address decode, row and column selection, and sense amplifier read and verify circuits would be implemented in the diffusion/substrate and some of the lower metal layers.

Key Parameter	Performance
Area	4-16 F^2
Bandwidth per array	4-8 bits
Read Latency	200-700 ns
Write Latency	1 us
Cell Leakage	0.1 nA/cell
Program Energy	10-100 pJ/cell
Endurance	> 10^5 - 10^8 cycles
Retention	> 7-10 years
Scaling Potential	< 10 nm
Ron/Roff ratio	100
Selectivity (DI @VR, VR/2)	> 10^6 - 10^{10}

Table 1: Crossbar 1S1R ReRAM Parameters

When the ReRAM memory circuits are placed alongside other blocks, the peripheral circuits would be realized as blocked areas. In traditional digital implementation flow, we can think of these as placement blockage areas on which standard cells cannot be placed. Crossbar's stacked ReRAM memory utilizes roughly a 25% memory to periphery area ratio for a 2-layer stack.

In a full-chip implementation, we expect multiple arrays to be distributed across the chip, as mentioned in section 1. This naturally causes several areas of "blocked" regions where the peripheral circuits of the ReRAM need to be placed. At the same time, it is desirable to allow for some signal connectivity through these blocked regions to alleviate routing congestion of the logic circuits. Because the peripheral area must allow for connection to the ReRAM layers, there will be restriction relating to the interconnect routing over these blocked regions. The specific metal layers that are blocked have significant impact on the routability of the overall integrated design. Completely blocking all metal routing over the blockage region necessitates any routing connections to go around the blocked regions which results in significant additional routing area overhead with an integrated design. A more realistic assumption would allow a limited amount of metal routing over the peripheral region for routing connections to pass through. This can be realized even if the number of ReRAM stack increases, as the blocked region could expand to accommodate a staggered connection from the higher memory metal layers to the base transistors below.

Figure 4 shows how it would be possible for a signal feed-through path through the blocked ReRAM peripheral logic region by staggering the via tap points from the ReRAM layer above. The width of the ReRAM peripheral circuit may need to increase as the number of stack layers increase. This is the approach we have adopted for our study.

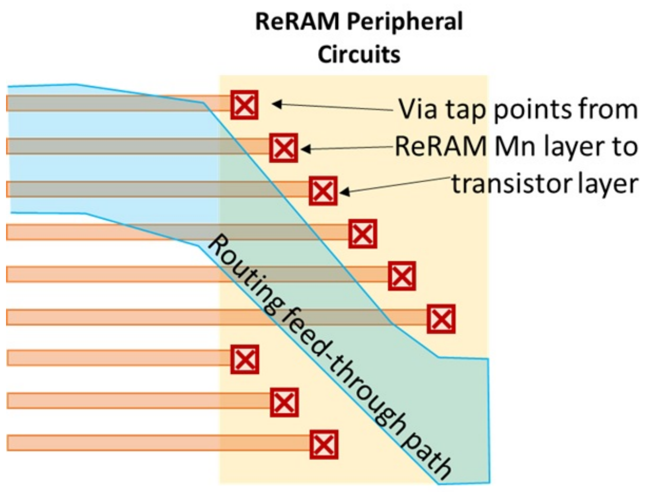

ReRAM Peripheral Circuits
Via tap points from ReRAM Mn layer to transistor layer
Routing feed-through path

Figure 4: Limited Signal Feed-through Paths

In our approach, we assume that ReRAM is placed in the upper most level, which is above metal-10 layer for the process technology we have used for this study. Signal feed-through paths across the blocked ReRAM peripheral circuits are critical to efficiently make use of noncontiguous floorplans. We had observed that when we completely block signal routing on the ReRAM peripheral circuit region, the Auto-Place-and-Route (APR) fails in generating the layout. The blockage layer settings we have used limits metal usage completely on lower layers but allows for signal feedthroughs on metal 9 and 10, just under the metal-11 layer used by the ReRAM array. Since details of the metal usage by the ReRAM is process-specific, these assumptions serve as a rough approximation of routing congestion faced during the APR step.

3 AREA STUDY OF RERAM-PROCESSOR INTEGRATION

Our aim for the physical design study was to explore a monolithic processor core that can be physically integrated with a ReRAM memory on the same chip. In this section, we present area experiments of integrating a standard-cell based synthesized RISC processor circuit with a ReRAM crossbar memory circuit and analyze the area and routing congestion that results from such an integration. Two different integration configurations are presented in this section with the area impact results obtained.

3.1 CAD Methodology

Our Digital Implementation CAD flow, shown in Figure 5, uses Synopsys Design Compiler for Synthesis and Cadence

Encounter to perform the Auto-Place-and-Route (APR) step of the flow to produce the final GDSII layout.

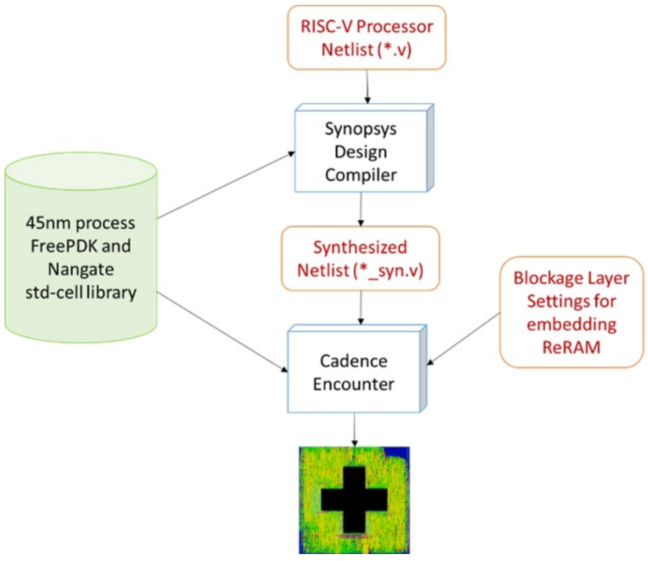

RISC-V Processor Netlist (*.v)
Synopsys Design Compiler
45nm process FreePDK and Nangate std-cell library
Synthesized Netlist (*_syn.v)
Blockage Layer Settings for embedding ReRAM
Cadence Encounter

Figure 5: CAD tool flow for Digital Implementation

We use the open-source NCSU FreePDK 45nm process design kit and the Nangate open source digital library for the 45nm process node. This allows us to perform relative comparison studies to model the physical integration of ReRAM with a processor. We chose the open-source Berkeley RISC-V VSCALE processor as the core for studying the processor-memory area impacts. The synthesizable Verilog netlist of the core is called VSCALE and uses a 32-bit instruction set with a single-issue in-order 3-stage architecture.

To mimic the integration constraints listed in the previous section (2.2), two types of blockage layers are indicated in the Cadence Encounter setting. The first is for the placement blockage to prevent standard cells from being placed, and the second is routing blockage for the specific metal layers to limit routing. For this work, we mimic the restricted metal routing described in the previous section by blocking metal layers 1-8 and allowing for the APR tool to route through the blocked region using metal 9 and 10. The ReRAM memory layers are assumed to be in metal layers 11 and 12 above.

3.2 Single ReRAM Cluster Integration

Our first study consisted of creating a physical layout of the integrated ReRAM peripheral circuit with the VSCALE core using blockage layers in the Cadence Encounter tool. The blockage settings and placement are based on the initial specifications provided by Crossbar on the integration constraints of an ReRAM array with the logic area underneath. The peripheral region is to be in the shape of an 'L' for the support circuitry, such as row and column decoders, sense amplifiers, and program circuits. We can implement this constraint in the digital implementation flow by creating a physical blockage region in the shape of an 'L' that prevents any standard cells from being placed. Given the L shape of the ReRAM's support circuitry, we have adopted a four-instance centrally located ReRAM configuration, with a cross shape, as shown in Figure 6.

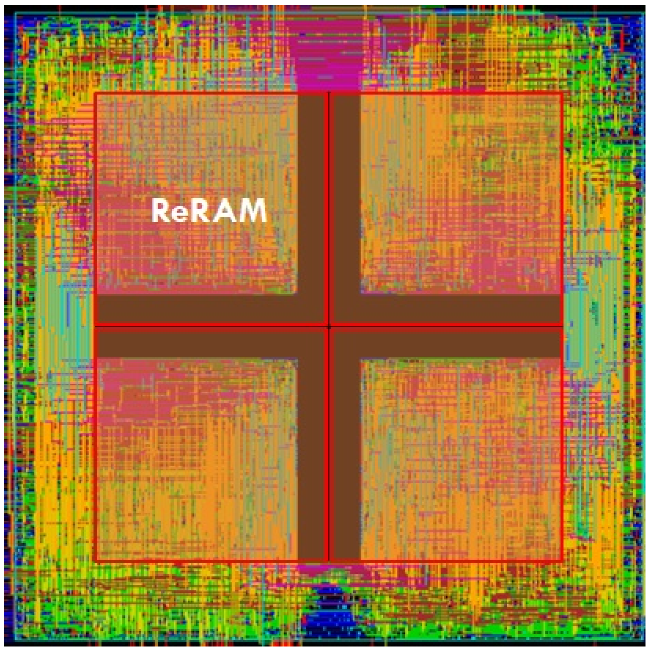


Figure 6: Digital Implementation of an integrated ReRAM RISC-V Processor

This configuration allows for the peripheral blockage regions to be abutted with each other, resulting in a contiguous blockage region for higher area utilization. This configuration has the advantage of allowing for I/O connectivity between the ReRAM memory and the processor, as well as allowing for connection between the overall tile which would need to communicate with other blocks.

We first used Cadence Encounter to perform the APR and generate the layout for the core alone. Our approach measures minimum feasible area by iteratively reducing the floorplan dimension and checking for congestion, DRC, connectivity violations. For the VSCALE core, using this PDK, the synthesized netlist targeted an operating frequency of 150MHz with a total of 59,672 standard cells at the 45nm process technology (nominal process, 1v, 27c). The VSCALE processor's area without any integration was found to be 30,373 sq um, after performing APR.

We next created a physical layout of the integrated ReRAM peripheral circuit with the VSCALE core using the above physical constraints as inputs to the Cadence Encounter tool. For this experiment, we used 4 ReRAM arrays, each of size 75um x 75um, which corresponds to a memory capacity of about 0.5MB for a 2-layer ReRAM stack. Note that crossbar has demonstrated feasibility of scaling to 8-layers for the ReRAM stack. Table 2 summarizes these results.

	Results
Standard-cell area	22,088 sq um
Target Frequency	100 MHz
Minimum Core Area for Processor Alone	172 um x 172 um = 30,373 um Core Density = 98.9% Total Wire Length = 358 um
Minimum Area for Integrated Version with Blocked area: 5,600 sq um	200 um x 200 um = 40,026 um Core Density = 84.8% Total Wire Length = 278 um
Additional Area Impact due to Integration	40026/(30373+5600) = 1.1127 -> 11.3% penalty

Table 2: Area Impact of Processor-ReRAM Integration

Since the peripheral region takes 25% of the ReRAM area, this amounted to a total blocked region of 5600 sq um for this configuration. Figure 6 shows the final APR layout with the VSCALE processor along with the metal layers. Each of the red-square represents a single ReRAM array (with the L-shaped peripheral region underneath). The total area of this integrated design was 40,026 sq um, which is 31% higher than the digital implementation of the processor alone. Note that this addition includes an additional area penalty from adding in the embedded block of 11.3% (after accounting for the blockage area and the area of the cells) in 45nm process.

The area penalty from the integration is measured as the difference between the total area of the integrated design and the sum of the VSCALE processor area and the ReRAM blocked region. This area penalty is mainly attributed to additional area needed for the routing of signals due to the blocked area in the center, around which there would be a higher incidence of routing congestion. There is also minor contribution due to standard-cell placement inefficiencies and additional filler cells incurred due to the larger overall area of the integrated block.

3.3 Multiple ReRAM Cluster Integration

The previous area study used a single ReRAM array to fit within the VSCALE core. VSCALE is a 32-bit integer core and is not representative of realistic cores which tend to be larger and more complex. For our next study, we considered a scenario where a larger processor is used, multiple ReRAM arrays are embedded for increased bandwidth. The scenario that we have setup for the second experiment is a 256-bit scaled version of the VSCALE processor which is integrated with four instances of the ReRAM crosses. Figure 7 shows the digital implementation of this design.

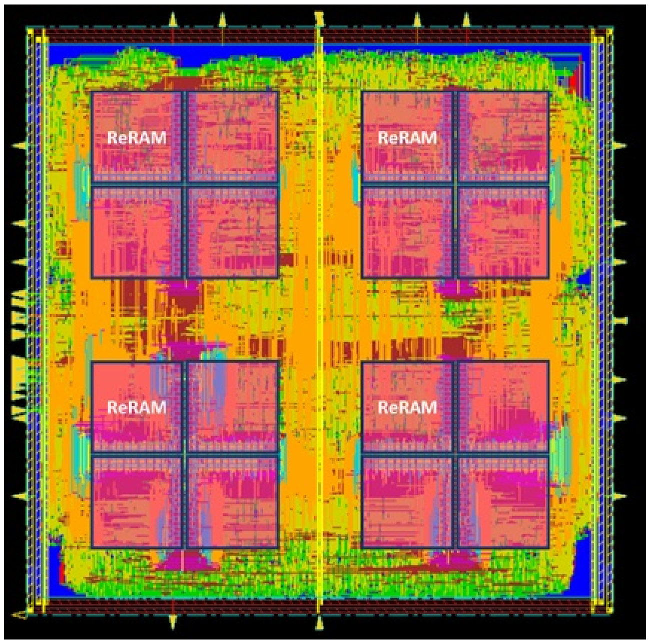
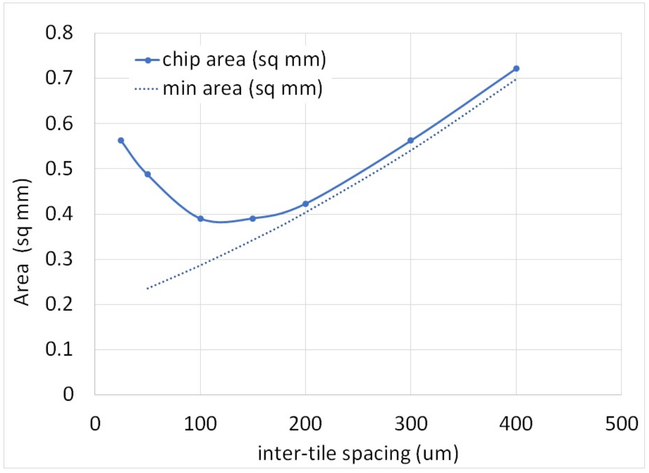


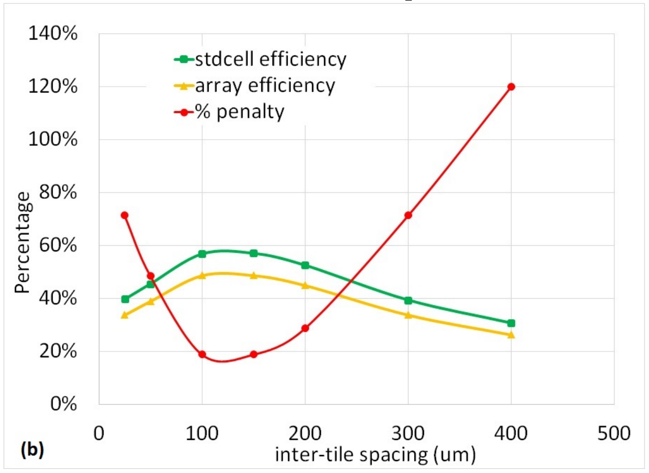
Figure 7: Multiple ReRAM clusters integrated with a 256-bit RISC-V Processor

Although this extrapolation only scales the data path portion of the processor and will not model impacts of the control path of a more complex, realistic processor, we can still infer the impact of routing congestion from a larger processor. The implementation shows a 2x2 array of ReRAM crosses integrated with a 256-bit integer RISC-V processor. Using a ReRAM array of size 109um x 109um, the total ReRAM data storage realized would be 4MB at 45nm process node for the 2-layer stack. In this study, we seek to find the impact of the spacing between the ReRAM crosses on the overall area efficiency. This was achieved by iteratively providing smaller floorplan dimensions at each spacing to find the minimum dimension. The results of the experiment are shown in Figure 8.

The design-area parameter reported in Figure 8 (a) is the minimum area that was feasible using our EDA flow and has successfully passed the DRC, LVS, and timing constraints. The min-area (dotted-line) shown on the graph is the theoretical limit on the minimum area possible considering the inter-tile spacing and the ReRAM embedded block size. At higher inter-tile spacing, this theoretical limit is the lower bound as the routing constraints don't restrict the layout. At the lower-end of the inter-tile spacings, we see the effect of the routing congestion limiting the feasible design-area.



(a) Overall Area Impact



(b) Area Efficiency Impact

Figure 8: Impact of Inter-ReRAM array spacing on Area and Efficiency

Figure 8 (b) reports the array and standard-cell efficiency values measured. Array efficiency is calculated by dividing the ReRAM array area by the minimum feasible design area to represent the percentage of usable ReRAM array. The theoretical limit for this would be 100%, if the entire chip could be covered by ReRAM array. The "stdcell efficiency" parameter reports the percentage of the total area used by the standard-cell and represents the amount of usable CPU area. The theoretical limit for this value would be 75%, as 25% would need to be blocked for the ReRAM's peripheral logic.

As shown in Figure 8 (b), an optimum inter-ReRAM spacing exists to maximize ReRAM array efficiency at close to 50%, at which point the area penalty is 18%. This area penalty follows the calculation given in section 3.2, as follows.

$$Area.Penalty = \frac{Integrated.Area}{(CPU.Logic.Area + ReRAM.Peripheral.Area)}$$

The results indicate that if there is not sufficient spacing between the ReRAM peripheral circuit's blocked regions, then network congestion occurs which cannot be resolved by the APR. On the other hand, if the spacing is too far apart, the entire generated layout can fit between the tiles resulting in large unused spaces. Note that it might be possible to optimize this layout manually and utilizing the areas in the corner to overcome this, however manual layout is beyond the scope of our initial area study. At 45nm, with our design configuration, the optimum inter- spacing is 100um to 150um. Larger spacing (> 200um) leads to inefficiency from unused synthesized areas (empty space) while smaller spacing (<100um) leads to inefficiency from routing congestion between standard cell groups. For alternative configurations, the specific optimal point could be affected by the relative size of the processor and the blocked region due to the ReRAM array and would be worth investigating this relationship in a future study.

Extrapolating these results to an 8-layer stack would create a 16MB ReRAM memory integrated into the ReRAM CPU tile with an area of 0.4 mm^2. For a 400 mm^2 die size, the above ReRAM array could be tiled 1000 times across the chip, to produce a total storage capacity of 16GB ReRAM. Because an 8-layer stack would require additional peripheral circuits to decode the wordline per stack and/or higher current driving transistors, the number of cores will be scaled down. At the 16nm process, assuming a 10x reduction in area, a 400 mm^2 chip should be capable of delivering 160GB ReRAM storage.

3.4 SRAM-ReRAM Integrations

One other configuration of interest is integrating an SRAM memory array underneath the ReRAM memory. The physical layout of an SRAM memory also follows a 'L' shaped peripheral region encompassing an array of SRAM cells. SRAM and ReRAM layouts complement each other perfectly because SRAM bitcells occupy the Front-End-Of- Line (FEOL) layers, while ReRAM layers occupy the BEOL layers. This means that for the SRAM array cells themselves, they only from substrate upto metal-2 layers leaving all of the above layers free and unused. On the other hand, for the ReRAM array cells, they would occupy higher metal layers, such as metal-11 and metal-12 in our 45nm examples. This allows them to be fitted naturally within each other and makes SRAM a ideal candidate to be placed underneath the ReRAM from a layout point of view.

From a system architecture point of view, we are interested in this SRAM-ReRAM configuration as an SRAM write-back cache to an ReRAM main-memory. This would minimize the impact of ReRAM write latency, which is on the order of 1us. For our study, we have selected an opensource academic memory compiler, called OpenRAM, created by UC-Santa Cruz and OSU [5]. This tool includes SRAM leaf cells for the 45nm process using the same FreePDK45 design kit used by our standard-cell logic.

Figure 9 shows four SRAM arrays placed together with ReRAM array on top. The SRAM arrays are rotated to allow for the I/O ports of the SRAM to be accessed externally and not conflict with the central control region of the ReRAM array. The total SRAM capacity in this instance is 16kB (4kB each SRAM) with a total layout area of 211,725 sq. mm.

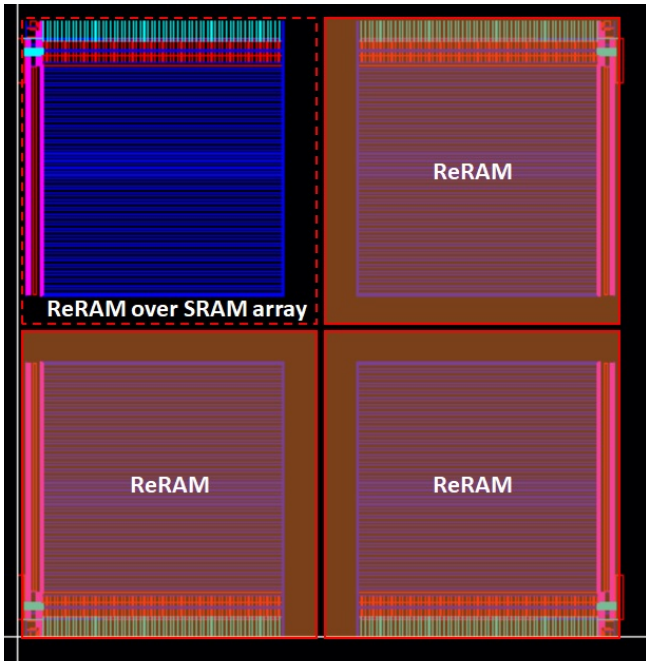


Figure 9: ReRAM Integrated with SRAM Memory

Note that the SRAM array generated in our study is made using an academic SRAM bitcell of size 1.344um x 0.707um, which is approximately 3x larger than industry SRAM bitcells at the 45nm process node. Therefore, the SRAM capacity can be expected to scale with a more realistic, smaller bitcell size and can be verified by using a commercial memory compiler instead of the OpenRAM memory compiler we used. However, the floorplan proposed for the SRAM-ReRAM configuration would still be applicable.

Compared to the ReRAM-CPU layout, SRAM's array region would largely be limited to the lower metal layers (below metal-4). Therefore, we feel that the ReRAM I/O connections can be made on the higher regions without difficulty. Also, because the four SRAM arrays are independent blocks, there is no need for the signal feedthroughs on the peripheral blockage regions, which makes this a more straightforward implementation.

4 ARCHITECTURAL SIMULATION

We also explored the performance implications of a ReRAM based main-memory architecture against a DRAM architecture, as well as the impact of a lower write-latency due to a reduced write-energy requirement. Figure 10 shows the architectures we used for comparing a ReRAM architecture with 1000 memory-access points against a DDR4 based DRAM architectures with six memory controllers. We used a mesh type of network-on-chip (NoC) topology for both architectures.

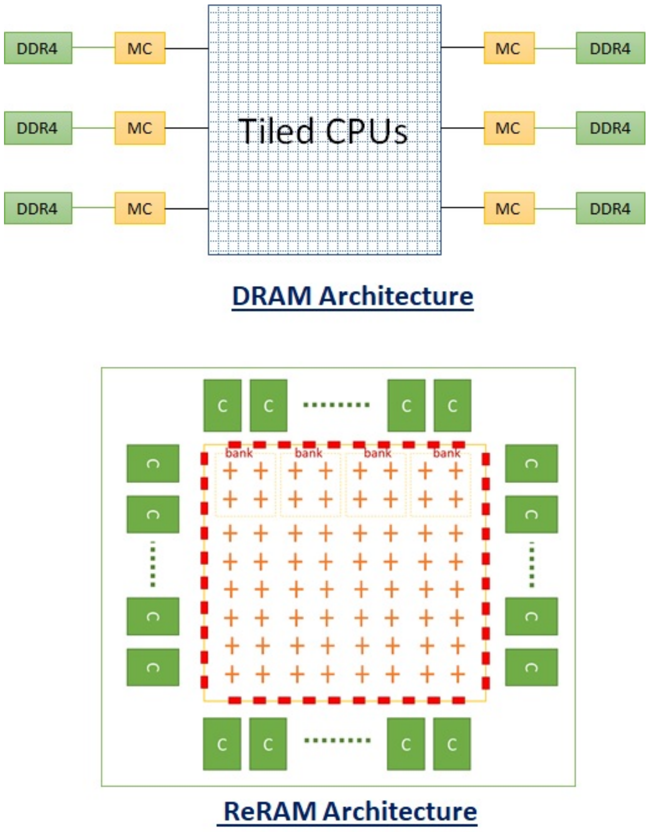


Figure 10: Architectural Comparison

We performed this analysis using an open-source architectural simulator, Structural Simulation Toolkit (SST) from Sandia National Laboratories [12]. SST is a component based, cycle-accurate simulator that is useful for fast comparison of different scenarios. Our CPU was modeled using Miranda, a pattern-based CPU simulator, to support 8 issues per core per cycle. We used the hardware-verified DRAMSIM3 simulator to model a dual-rank DDR4-2666 DRAM device operating at

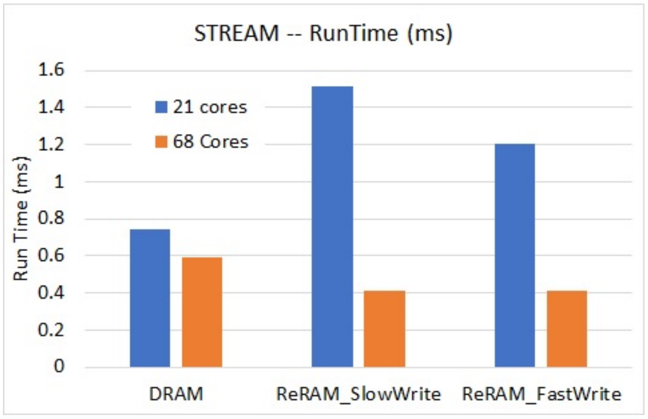
2.66GHz. For ReRAM, we assumed a centrally located memory IP with support controllers and bank-select logic located underneath the memory, while the CPUs surround the array. We used two benchmarks to compare our architecture: STREAM and GUPS. STREAM mimics dense memory access and is expected to favor the higher access granularity of traditional DRAM architectures. GUPS mimics sparse memory accesses for graph like algorithms and was expected to favor ReRAM's low-access granularity. Based on our area analysis, we expected several mini ReRAM tiles to be accessed in parallel to provide sufficient bandwidth.

Figure 11 summarizes the result of the architectural simulation. In order to understand the impact of the longer write latency of ReRAM, we compared DRAM with two versions of ReRAM: SlowWrite and FastWrite. For the ReRAM SlowWrite version, we used a write-latency of 1us, while for the FastWrite version, we used 200ns. The read latency was set to 200ns for both versions of the ReRAM. We simulated the comparison with both 21 cores and 68 cores, based on expected number of cores that could be fabricated on a standard chip size. The results indicate that when the number of cores is low (21), DRAM-based architecture outperforms ReRAM, even for GUPS type of algorithms. Although there was a small performance improvement with the ReRAM-FastWrite version, this still was not enough to overcome DRAM architecture performance. However, when the number of cores was increased from 21 to 68 cores, we see that in both STREAM and GUPS based benchmarks, ReRAM is able to outperform DRAM-based architecture. This is due to the higher amount of memory access requests needed with the higher core count. This requirement is more easily met by a more parallel memory system such as the one architected with the ReRAM based main-memory. There needs to be enough parallel request to fully exploit the high amount of parallelism afforded by ReRAM and overcome the higher latency with ReRAM.

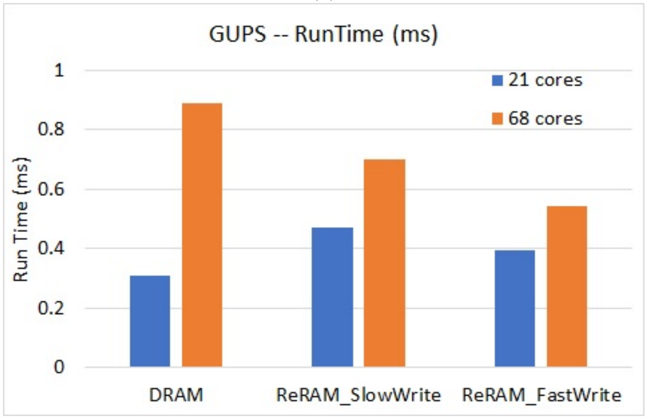
Table 3 compares the memory bandwidth processed in each of the simulated conditions. At lower core count, DRAM-based architecture provides STREAM bandwidth of 76GB/s is nearly 40% higher than the one provided through ReRAM-based architecture. At higher core count, ReRAM provides a higher bandwidth of 138GB/s, while the DRAM-based architecture's STREAM bandwidth is 30% lower at 95GB/s.

Using the architectural simulation framework, we next simulated for a number of core count to compare ReRAM against a DRAM-DDR4 and a High-Bandwidth-Memory-2 (HBM2) version of DRAM main memory. The results, shown in Figure 12, confirm that with sufficient processing power, the highly parallel ReRAM with long latencies performs better than high-speed DRAM with limited memory controllers.

The cross-over point when ReRAM outperforms is 85GB/s for DRAM-DDR4 and 135GB/s for DRAM-HBM2 device with the STREAM benchmark. One interesting note is with the slight worsening of performance with HBM DRAM at very high core count of 512. Because of HBM's higher bandwidth interface, the low-access granularity of GUPS suffers with HBM due to stalls from prior access requests. In the STREAM bandwidth plot, the star represents the reported 90+ GB/s number from Intel Knights Landing (KNL) [13], which corroborates with our simulation results. At this point, ReRAM



(a)



(b)

Figure 11: Impact of Inter-ReRAM array spacing on Area and Efficiency

Bandwidth (GB/s)	Cores	DRAM	ReRAM _SlowWrite	ReRAM _FastWrite
GUPS	21	1.36	0.89	1.07
	68	1.53	1.94	2.51
STREAM	21	76	37.45	47.07
	68	95.6	136.63	138.6

Table 3: Bandwidth Comparison

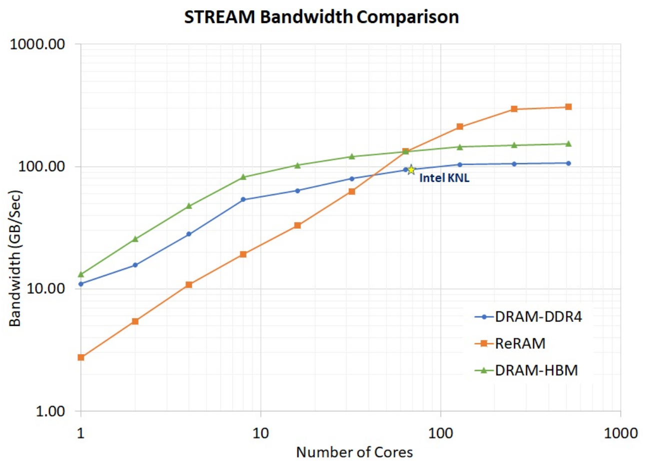


Figure 12: STREAM Benchmark Comparison Results

outperforms DRAM-DDR4 by 30% for the STREAM benchmark case and meets the performance of HBM2-based architecture.

5 CONCLUSION

In this work, we have demonstrated a method for evaluating integrated ReRAM-Processor type of architectures using standard EDA tools. We also presented an overview of Crossbar ReRAM technology that has been demonstrated in fabricated silicon chips that allow this novel on-chip main-memory architecture. Our layout results indicate that we can integrate a cluster of ReRAM arrays with a processor logic underneath and incur an area penalty of 18% and an overall area efficiency of 50%. Finally, architectural simulations comparing ReRAM and DRAM based architectures showed that ReRAM-based main memory architectures outperforms at higher core counts, where their high amount of memory parallelism can be sufficiently utilized.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under Award No. 1642424 and the Department of Defense under Contract No. FA8075-14-D-0002-0007, TAT 15-1158.

REFERENCES

[1] M. Jagasivamani, C. Walden, D. Singh, L. Kang, S. Li,M. Asnaashari, S. Dubois, B. Jacob, and D. Yeung, "Memory-systems challenges in realizing monolithic computers", Proceedings of the International Symposium on Memory Systems - MEMSYS 18, 2018.
[2] ReRAM Memory | Crossbar. (n.d.). Retrieved from https://crossbar-inc.com/en/
[3] Y. Chen, C. Petti, "ReRAM technology evolution for storage class memory application," 2016 46th European Solid-State Device Research Conference (ESSDERC), Lausanne, 2016, pp. 432-435.
[4] Sung Hyun Jo, T. Kumar, S. Narayanan, W. D. Lu and H. Nazarian, "3D-stackable crossbar resistive memory based on Field Assisted Superlinear Threshold (FAST) selector," 2014 IEEE International Electron Devices Meeting, San Francisco, CA, 2014, pp. 6.7.1-6.7.4.
[5] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, M. Sarwar, "OpenRAM: An Open-Source Memory Compiler", Proceedings of the 35th International Conference on Computer-Aided Design (ICCAD), 2016.
[6] I. Bhati, M. T. Chang, Z. Chishti, S. L. Lu and B. Jacob, "DRAM Refresh Mechanisms, Penalties, and Trade- Offs", in IEEE Transactions on Computers, 2016, vol. 65, no. 1, pp. 108-121.
[7] John L. Hennessy, David A. Patterson, Computer Architecture: A Quantitative Approach, Elsevier, pp. 289, 2007.
[8] T. Y. Liu et al., "A 130.7mm2 2-layer 32Gb ReRAM memory device in 24nm technology", 2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers, San Francisco, CA, 2013, pp. 210-211.
[9] R. Fackenthal et al., "19.7 A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology", 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco, CA, 2014, pp. 338-339.
[10] S. I. Association, "International technology roadmap for semiconductors," in ITRS Report, 2017.
[11] Lei Wang, CiHui Yang, Jing Wen, and Shan Gai, "Emerging Nonvolatile Memories to Go Beyond Scaling Limits of Conventional CMOS Nanodevices", Journal of Nanomaterials, vol. 2014, Article ID 927696, 10 pages, 2014.
[12] A. F. Rodrigues, R. C. Murphy, P. Kogge, and K. D.Underwood, "Poster reception: the structural simulationtoolkit",Proceedings of the 2006 ACM/IEEE conference onSupercomputing - SC 06, 2006.
[13] J. Jeffers, J. Reinders, and A. Sodani, "Knights landingoverview",Intel Xeon Phi Processor High Performance Programming, pp. 15-24, 2016.
[14] Xu, C., Niu, D., Muralimanohar, N., Balasubramonian, R., Zhang, T., Yu, S., Xie, Y., "Overcoming the challenges of crossbar resistive memory architectures", 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). doi:10.1109/hpca.2015.7056056
[15] Zhang, H., Xiao, N., Liu, F., Chen, Z., "Leader: Accelerating ReRAM-based Main Memory by Leveraging Access Latency Discrepancy in Crossbar Arrays", Proceedings of the 2016 Design, Automation and Test in Europe Conference and Exhibition (DATE).