DRAMA: An Architecture for Accelerated Processing near Memory

Amin Farmahini-Farahani[†], Jung Ho Ahn[‡], Katherine Morrow[†], and Nam Sung Kim[†]
[†]University of Wisconsin-Madison, [‡]Seoul National University

Abstract—Improving energy efficiency is crucial for both mobile and high-performance computing systems while a large fraction of total energy is consumed to transfer data between storage and processing units. Thus, reducing data transfers across the memory hierarchy of a processor (i.e., off-chip memory, on-chip caches, and register file) can greatly improve the energy efficiency. To this end, we propose an architecture, DRAMA, that 3D-stacks coarse-grain reconfigurable accelerators (CGRAs) atop off-chip DRAM devices. DRAMA does not require changes to the DRAM device architecture, apart from through-silicon vias (TSVs) that connect the DRAM device's internal I/O bus to the CGRA layer. We demonstrate that DRAMA can reduce the energy consumption to transfer data across the memory hierarchy by 66-95% while achieving speedups of up to 18× over a commodity processor.

1 Introduction

A major source of energy inefficiency in current computing systems originates from the fact that the processor usually must transfer data from off-chip DRAM devices to its on-chip cache hierarchy and then into its register file before processing it. Our experiment shows that evaluated (data-intensive) applications consume up to 66% of their combined memory I/O and processor energy just for moving data from off-chip memory to the last level cache and through the cache hierarchy, buses, and register file in 40nm technology generation. Furthermore, the fraction of data movement energy in total system energy is projected to increase even further with technology scaling [3].

In order to minimize such energy inefficiency, this paper makes the following contributions:

- We propose a new DRAM-Accelerator (DRAMA) architecture where the processor can offload compute- and/or data-intensive operations to CGRAs 3D-stacked atop DRAM devices (Section 2). This can reduce data transfers across the conventional processor memory hierarchy (i.e., from the off-chip DRAM to on-chip caches and register file) and thus energy consumption. Moreover, CGRAs exploit spatial parallelism in applications to significantly improve performance. Apart from added TSVs on a DRAM device that connects its internal I/O bus to the CGRA layer, DRAMA does not change the processor, the processor-memory interface, and the underlying DRAM architecture. This is particularly attractive to DRAM manufacturers exploring features that can boost their product values.
- We provide a detailed methodology on how to partition application data and place them on DRAM devices such that we can maximize the benefit of DRAMA architecture executing accelerated applications while maintaining the compatibility with the existing dual-inline memory module (DIMM) architecture and its interface (Section 3).

Our experiments for two classes of applications ported to DRAMA show that the energy consumption to transfer data across the memory hierarchy is reduced by 66-95% while achieving speedups of up to 18× over a commodity processor. The energy reduction in transferring data translates into a 40-66% reduction in the total processor and memory I/O energy consumption which is achieved by stacking CGRAs atop DRAM devices.

Previous studies have examined integrating logic with memory (e.g., [9]), but differences in logic and memory technologies caused these processor-in-memory (PIM) systems to suffer from high manufacturing costs and low yield [10]. Alternatively, 3D stacking technology can provide a high-bandwidth connection between DRAM and processor dies using TSV-based wide I/O buses. Although there have been proposals on stacking accelerator layers and DRAM layers [13], thermal effects limit the number of 3D-stacked DRAM and logic layers (reducing capacity) [7]. In contrast, DRAMA instead stacks a single layer of a small, low-power CGRA atop each single-layer DRAM device in a DIMM. Thus, DRAMA has a limited negative impact on thermal reliability.

2 Hardware Architecture

The DRAMA architecture stacks a CGRA on top of each DRAM device, connected to the internal DRAM I/O lines using TSVs. A conceptual view of the architecture is shown in Figure 2a. Each CGRA is connected only to its associated DRAM device and operates on the data contained in that device independently of (and in parallel with) the CGRAs on the other DRAM devices. DRAMA is designed to conform with the conventional DIMM interface; it does not require any change to the DIMM interface or the processor while involving minimal changes to the underlying DRAM design. As a result, DRAMA can be easily used in conjunction with existing processors and their platforms to accelerate DRAMA-enhanced applications. Furthermore, this design still allows existing un-accelerated applications to run on DRAMA-equipped platforms without performance penalty.

The processor communicates with CGRAs through a memory-mapped I/O interface that operates similarly to mode registers in conventional DRAM systems, and that does not require any changes to the processor-DIMM interface. The processor sends kernel configuration data, address generation parameters for the data to be processed, and other

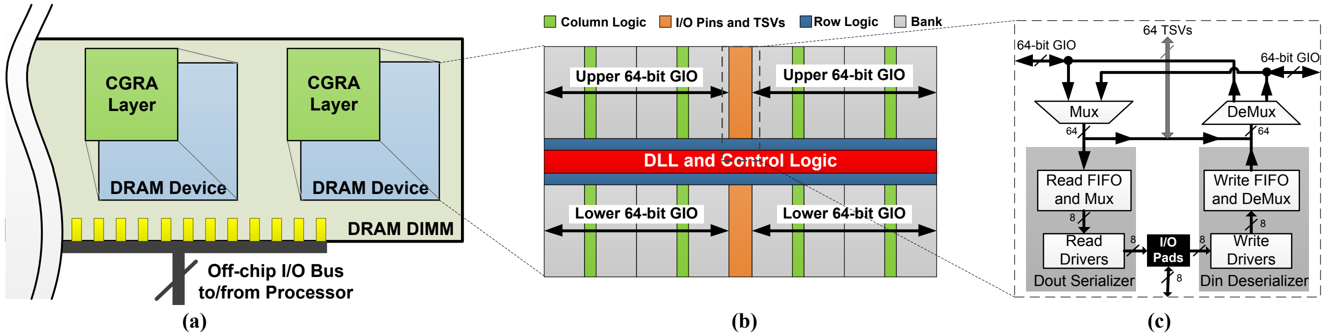

Figure 2. Organization of the DRAMA architecture (a), internal architecture of a DDR3 device with ×16 interface and 8 banks (b), TSV connection to upper 64 global I/O lines (c).

kernel parameters before triggering kernel execution using this interface. The processor polls a memory-mapped status register to check for kernel completion.

2.1 CGRA – DRAM Device Interface

Figure 2b shows the architecture of a modern DRAM device with an ×16 interface (16-bit data I/O per DRAM device) [8]. The device has eight banks divided into two groups (left, right) of four banks. Each bank group shares a 128-bit inter-bank global I/O (GIO) bus comprised of upper and lower 64-bit GIO buses. For each read operation, multiplexers select 128-bit data from the left or right 128-bit GIO buses; this data is then serialized before being sent through the 16-bit data I/O pins (Figure 2c).

The CGRA layer simply reads or writes data through the TSVs connected to the existing GIO buses (Figure 2c) without changing the underlying architecture of the DRAM device. DRAM devices with ×8 and ×16 interfaces use 64 and 128 TSVs to transfer between the DRAM and CGRA, respectively. This TSV-based stacking approach is similar to one developed for stacking a wide-I/O DRAM device atop a processor die [4]. In DRAMA, the same steering logic that directs data to/from the left or right set of banks also directs data through the TSVs to the CGRA layer. Additional TSVs transfer command and address bits from the CGRA layer to the DRAM device to indicate the type of operation and the address involved (which in turn determines the bank, row, and column within the DRAM device). Therefore, data is transferred between the CGRA and DRAM device using normal DRAM operations. Each CGRA connects to the DRAM device through a small amount of input/output buffering and logic that then connect to the TSVs (Figure 1).

Current 3D interconnect technology allows a TSV pitch size of 50µm for 3D DRAM stacking [4], thus 128 TSVs have an area of only ~0.32 mm^2. Since the typical area of DDR3 and DDR4 devices ranges from 30 mm^2 to 80 mm^2 [6], the area overhead of the additional TSVs is negligible. This TSV-based stacking provides the same peak bandwidth per device as the off-chip data I/O bus connection.

2.2. CGRA Architecture

CGRAs exploit spatial parallelism to accelerate a wide range of applications. Each CGRA is a heterogeneous grid of 32-bit coarse-grained functional units (FUs) and small distributed storage (for intermediate results) connected by a configurable routing switches. Based on synthesis results of FUs and switches, we estimate that the area of a CGRA with 64 FUs is ~0.832 mm^2 in 40 nm technology targeted at 800 MHz.

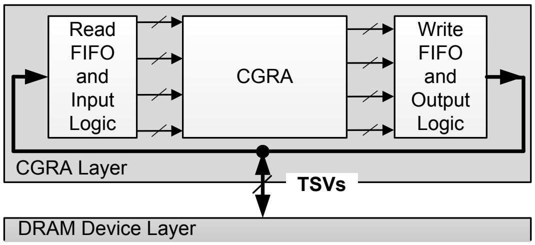

Figure 1. DRAMA CGRA stacked on top of DRAM Device.

Each CGRA also includes a simple memory controller (MC) to issue memory requests to its DRAM device. This CGRA-side MC does not require complex scheduling queues. Generally, accelerated kernels include little or no data-dependent control flow and avoid “pointer chasing”, so the memory request order can usually be optimized during kernel development. When CGRAs are launched, the control of DRAMs is passed to CGRA-side MCs and the processor-side MC stops sending commands like ACT and CAS. The processor and CGRAs do not operate on the same data set simultaneously. When CGRAs are working, the processor busy-waits until CGRA computations are completed to avoid frequent concurrent accesses. Thus, no changes to memory consistency are needed. If the processor-side MC requires sending memory commands, it first needs to halt CGRA-side MCs (by writing to a mode register). CGRA-side MCs then close pages before the processor-side MC takes control back. Next, the processor-side MC activates the required page and accesses data. The same mechanism is used for refreshing.

CGRAs in DRAMA use physical memory addressing, and usually operate on big data structures (like large images and matrices). Big-memory workloads rarely benefit from virtual memory features such as page swapping [1]. We adopt memory segmentations without paging for region of memory accessed by CGRAs. This approach maps contiguous region of virtual memory to contiguous region of physical memory. This eliminates virtual memory overheads due to TLB misses for the large data structures accessed by CGRAs in DRAMA.

3 Software Modifications

The conventional DIMM architecture interleaves each 64-bit data block amongst DRAM devices, each of which stores 4, 8, and 16 bits for ×4, ×8, and ×16 DRAM devices, respectively. However, DRAMA requires that all data used by each CGRA be located within the DRAM device to which that CGRA is attached. Maintaining compatibility with the existing DIMM

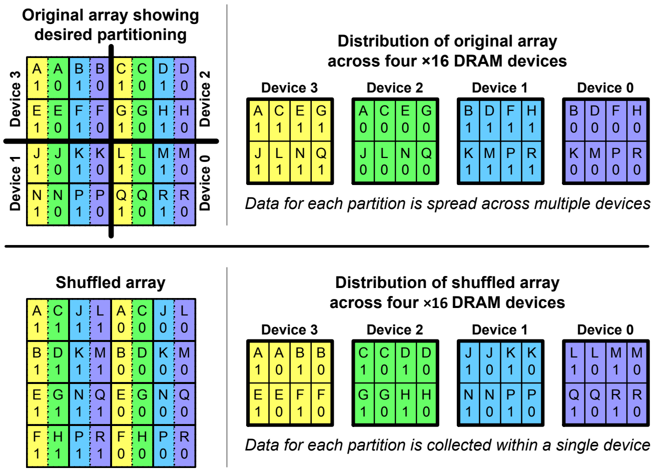

Figure 3. Data arrangement across devices for a 4×4 array of 32-bit words (A – R). Original data arrangement (top) and shuffled arrangement for DRAMA acceleration (bottom).

architecture and interface requires rearranging data both to partition it across CGRAs for parallel processing and to ensure that all sub-words of a given data word are written to the same DRAM device.

Figure 3 (top-left) shows an example of a 4×4 array of 32-bit words (A–R), where the four quadrants of the array should be distributed among the four ×16 DRAM devices in a DIMM as indicated. Each 32-bit data word consists of 16-bit upper and lower subwords, labeled "1" and "0", respectively. Figure 3 (top-right) shows how this array would typically be distributed across ×16 DRAM devices, separating the upper and lower halves of each 32-bit word, and failing to group the data for each quadrant into a single device. Figure 3 (bottom) shows a "shuffled" arrangement of the same data that partitions data appropriately for DRAMA processing when it is written to the DRAM using the conventional interface. The shuffling can be performed by the use of modified data structures, or by explicitly writing shuffled data during data structure initialization at the beginning of a program. To avoid inconsistencies between the processor's cache and main memory, CGRAs' shuffled input data is stored to memory using *non-temporal instructions*, which bypass the cache hierarchy (and avoid the need to flush the cache). After DRAMA execution, the processor "unshuffles" data produced by the CGRAs before using it.

4 Evaluation

We evaluate the efficacy of DRAMA executing embedded applications (SIFT, Tracking (TRCK), and Disparity map (DISP) from the San Diego Vision suite [12]), as well as scientific applications (LBM and MRI-Gridding (MRIG) from the Parboil suite [11]). The most compute- and data-intensive portions of each application are converted to dataflow graphs representing CGRA kernels (the remaining parts are executed by the processor). The energy consumption of each kernel is based on the number and type of computations in its graph. Kernel latencies are based on the depth and type of computations in the dataflow graph after it is mapped to the CGRA functional units. For all presented comparisons, DRAMA CGRAs have 64 FUs and operate at 800 MHz.

Baseline architectures without CGRAs are summarized in Table 1. For scientific application comparisons, the baseline is a high-performance four-way processor with DDR3-1600 DRAMs; for embedded application comparisons, the baseline is an embedded two-way processor with LPDDR2-1066 DRAMs. In each comparison, DRAMA uses the same processor and memory parameters as the corresponding baseline. For DRAMA and baseline architectures, we use DDR3-1600 devices with ×8 and ×16 interfaces and LPDDR2-1066 devices with an ×16 interface. To form a 64-bit bus between processor and memory, there exist eight and four DRAM devices for ×8 and ×16 interfaces, respectively.

TABLE 1. KEY ARCHITECTURE PARAMETERS

Baseline Architecture		
Feature	Scientific Comparisons	Embedded Comparisons
Processor Frequency	2 GHz	1 GHz
Superscalar	4-way	2-way
ROB/IQ/LSQ Entries	128/36/80	40/12/22
Integer / FP ALUs	3 / 1	2 / 1
L1I/L1D/L2 Size (KB)	32/32/512	32/32/512
Memory	DDR3-1600 (×8 and ×16)	LPDDR2-S4-1066 (×16)

DRAM Design		
Feature	×8 Interface	×16 Interface
# Devices (# CGRAs)	8	4
DRAMA TSVs Per Device	64	128

We extend *gem5* [2] and use McPAT [5] to evaluate the performance and power consumption of the DRAMA architecture, respectively. To estimate area and power of CGRAs and kernels executing on them, we synthesize CGRA's FUs and switches using Synopsys Design Compiler.

4.1 Performance

Figure 4 demonstrates the significant performance improvements of DRAMA for both scientific and embedded applications over the corresponding baseline architectures. This is the case even when comparing to a baseline with a doubled clock frequency ("Baseline 2x Freq"). For scientific applications, eight CGRAs each using a 64-TSV connection provide better speedup than four CGRAs each using a 128-TSV. In this case, the increased compute ability of additional CGRAs outweighs the increased bandwidth. In embedded applications, DISP achieves a speedup of over 18× due to regular memory access pattern and simple data flow graphs with high spatial parallelism. Overall, DRAMA's high speedups originate from parallel execution of CGRAs, spatial parallelism exploited by CGRAs, and direct data communication between CGRAs and DRAM devices (i.e., lower data communication latency for applications with low-temporal lo-

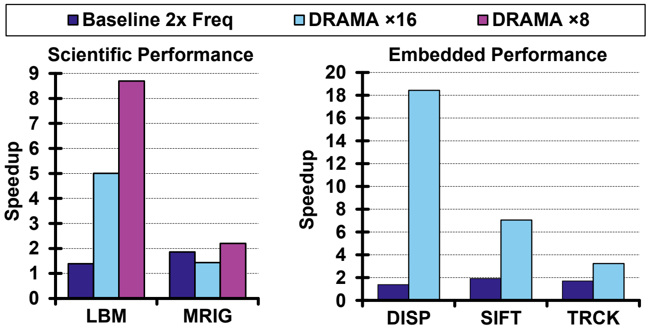

Figure 4. Speedup of DRAMA compared to the baseline architecture.

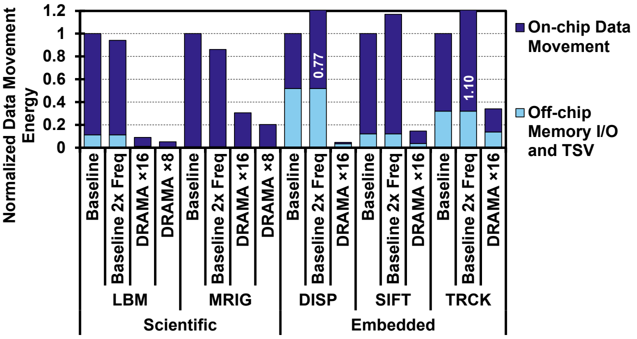

Figure 5. Data movement energy in DRAMA normalized to that in the baseline architecture.

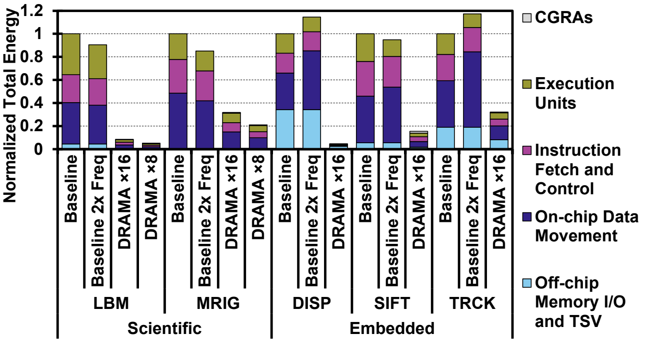

Figure 6. DRAMA (combined processor, CGRAs, and memory I/O) energy normalized to the baseline architecture energy.

cality due to not moving data across the cache hierarchy).

4.2 Energy Consumption

To evaluate energy consumption of DRAMA, we consider dynamic and leakage energy of the processor, cache hierarchy, and CGRAs. When CGRAs are working, the processor polls CGRA status registers at 1/10th of the processor's baseline frequency. Figure 5 shows the energy consumed for transferring data across the register file, cache hierarchy, and memory I/O in DRAMA normalized to that in the baseline architecture; DRAMA significantly reduces energy consumed for transferring data by 66-95%. In brief, DRAMA can greatly reduce data movement energy because of (1) processing most of data near memory devices and therefore eliminating data movement in the cache hierarchy and register file, and (2) using TSVs that consume 11.2× less I/O energy than off-chip interconnects in LPDDR2 [4].

Figure 6 shows the energy breakdown of DRAMA (combined processor, CGRAs, memory I/O), normalized to energy consumption of the baseline architecture. Processor energy is broken down into three parts of (1) instruction fetch and control, (2) execution units, and (3) on-chip data movement across the cache hierarchy, buses, and register file. DRAMA reduces total energy consumption over that of the baseline architecture by 68-95%. 34-60% of this energy reduction is achieved by using CGRAs and 40-66% of this reduction is achieved by stacking CGRAs atop DRAM devices. The DRAMA architecture with eight DRAM devices (hence eight CGRAs) provides more energy savings than the DRAMA architecture with four DRAM devices. Although eight CGRAs have higher combined CGRA leakage energy than four CGRAs, the leakage energy decreases due to shortened execution time. DISP executed by four CGRAs obtains the highest total energy reduction of 95%. The main reason for this significant energy reduction is that data movement energy in DISP is reduced by 95% which constitutes 66% of its total processor and memory I/O energy consumption.

Overall, DRAMA can substantially reduce energy consumption. Due to absence of the instruction fetch and control overhead, CGRAs process data much more energy-efficiently than processors. Moreover, unnecessary data movement is eliminated by stacking CGRAs atop DRAM devices. Finally, CGRAs speed up execution, reducing system leakage energy.

5 Conclusion

In this paper, we proposed a new architecture, DRAMA, to enable accelerated data processing near memory. DRAMA exploits local data processing to reduce data movement energy. DRAMA can be implemented using the current 3D stacking technology without changing the underlying DRAM architecture. We showed that DRAMA can considerably reduce energy consumption (68-95%) and improve performance by up to 18.4×.

Acknowledgments

This work was supported in part by generous grants from NSF (CCF-0953603, CNS-1217102, and CNS-0952425), and DARPA (HR0011-12-2-0019). Nam Sung Kim has a financial interest in AMD and Samsung Electronics.

References

[1] A. Basu et al., "Efficient virtual memory for big memory servers," in *International Symposium on Computer Architecture*, 2013, vol. 41, no. 3, pp. 237–248.

[2] N. Binkert et al., "The gem5 simulator," *SIGARCH Comput. Arch. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[3] S. W. Keckler et al., "GPUs and the Future of Parallel Computing," *IEEE Micro*, vol. 31, no. 5, pp. 7–17, Sep. 2011.

[4] J.-S. Kim et al., "A 1.2 V 12.8 GB/s 2 Gb Mobile Wide-I/O DRAM With 4x128 I/Os Using TSV Based Stacking," *IEEE J. Solid-State Circuits*, vol. 47, no. 1, pp. 107–116, Jan. 2012.

[5] S. Li et al., "The McPAT Framework for Multicore and Manycore Architectures," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 1, pp. 1–29, Apr. 2013.

[6] K.-N. Lim et al., "A 1.2V 23nm 6F2 4Gb DDR3 SDRAM with local-bitline sense amplifier, hybrid LIO sense amplifier and dummy-less array architecture," in *IEEE Intl. Solid-State Circuits Conference*, 2012, pp. 42–44.

[7] G. H. Loh et al., "Efficiently enabling conventional block sizes for very large die-stacked DRAM caches," in *Intl. Symp. on Microarchitecture*, 2011, pp. 454–464.

[8] C. Park et al., "A 512-Mb DDR3 SDRAM Prototype With CIO Minimization and Self-Calibration Techniques," *IEEE J. Solid-State Circuits*, vol. 41, no. 4, pp. 831–838, Apr. 2006.

[9] D. Patterson et al., "A case for intelligent RAM," *IEEE Micro*, vol. 17, no. 2, pp. 34–44, 1997.

[10] D. Patterson et al., "Intelligent RAM (IRAM): the industrial setting, applications, and architectures," in *Intl. Conf. on Computer Design*, 1997, pp. 2–7.

[11] J. Stratton et al., "Parboil: A revised benchmark suite for scientific and commercial throughput computing," 2012.

[12] S. K. Venkata et al., "SD-VBS: The San Diego Vision Benchmark Suite," in *International Symposium on Workload Characterization*, 2009, pp. 55–64.

[13] Q. Zhu et al., "A 3D-stacked logic-in-memory accelerator for application-specific data intensive computing," in *IEEE Intl. 3D Systems Integration Conf.*, 2013, pp. 1–7.