Near-DRAM Acceleration with Single-ISA Heterogeneous Processing in Standard Memory Modules

Hadi Asghari-Moghaddam[†], Amin Farmahini-Farahani[‡], Katherine Morrow[‡], Jung Ho Ahn[*], and Nam Sung Kim[†]

[†]University of Illinois at Urbana-Champaign, [‡]University of Wisconsin-Madison, [*]Seoul National University

Abstract

Energy consumed for transferring data across the processor memory hierarchy constitutes a large fraction of total system energy consumption, and this fraction has steadily increased with technology scaling. In this article, we present a near-DRAM acceleration (NDA) architecture wherein light-weight processors (LWPs) with the same ISA as their host processor are 3D-stacked atop commodity DRAM devices in a standard memory module to efficiently process data. In contrast to previous architectures, our NDA architecture requires negligible changes to commodity DRAM device and standard memory module architectures. This allows our NDA to be more easily adopted in both existing and emerging systems. Our experiments demonstrate that, on average, our NDA-based system consumes almost 65% lower energy at nearly 2× higher performance than the baseline system.

Keywords

DRAM, accelerator, near-data processing, die stacking.

1. Introduction

Energy inefficiency in current systems mainly originates from transferring data between where it is stored and where it is processed. With technology scaling, the fraction of data transfer energy in the total system energy has steadily and significantly increased [1]. Therefore, energy consumed for data transfers becomes much higher than actual computations.

To reduce data transfer energy, we can decrease the distance of data transfers by processing data near or in memory. This motivates us to re-examine the previous processing-in-memory (PIM) architectures (e.g., [2]) exploiting low access latency and high aggregate bandwidth enabled by integrating processor logic and DRAM on the same die. Such PIM architectures, however, suffered from (1) high manufacturing complexity; (2) poor processor logic performance; (3) large DRAM area per bit; and (4) design/verification challenges associated with custom DRAM architectures.

Recently, 3D-stacking has emerged as a promising integration technology. It can solve some of the critical problems faced by the previous PIM architectures because it integrates separate logic and DRAM layers with high-bandwidth low-energy through silicon vias (TSVs). Leveraging 3D-stacking technology, researchers have proposed near-DRAM acceleration (NDA) architectures that integrate accelerator logic and custom 3D DRAM devices to reap the benefits of both accelerators and near-memory processing (e.g., [3, 4]). More specifically, they focus on either accelerator architecture where its memory system is separate from the host processor's main memory system (similar to a discrete GPU architecture) [3] or the integration of accelerators and DRAM [4].

In this article, we stack a die of light-weight low-power processors (LWPs) with the same ISA as the host processor atop commodity 2D DRAM dies. It can be seamlessly integrated with the host processor's main memory system based on standard off-chip memory modules which can provide much larger capacity than other memory architectures such as high-bandwidth memory (HBM) for running traditional non-NDA applications. We first present an NDA architecture that requires no change to the host processor design and minimal changes to the commodity DRAM device's I/O circuitry while maintaining the compatibility with the standard DDR interface and DIMM architecture. Second, we explore three NDA microarchitectures that can provide diverse DRAM-accelerator bandwidth using commodity DRAM devices. Lastly, we evaluate the performance and energy-efficiency of our NDA architecture after discussing the benefit of 3D-stacking LWPs atop DRAM devices, compared with our prior architecture integrating reconfigurable accelerators [5].

2. Background

2.1 Conventional DRAM Architecture

The internal architecture of DRAM and its I/O circuitry is designed to facilitate high-bandwidth, low-cost connections to the accelerators stacked atop DRAM devices using TSVs. Figure 1(a) depicts a DDR3 DRAM device with a ×8 interface (8-bit data I/O per device), which consists of eight banks, each having two sub-banks split by the middle control logic. Except sharing common resources such as I/Os and DLLs, each bank operates independently of others. Each sub-bank is divided into multiple two-dimensional mats. The size of a mat is determined by the latency constraints of DRAM operations and the sensing ability of voltage difference induced by charges in a cell. This hierarchical structure of both control (decoders, wordlines, and column-select lines) and datapath (bitlines and local/global I/Os) within a bank is critical to improve area efficiency, bandwidth, and latency. Sense amplifiers are selectively utilized to speed up data delivery.

The eight banks are divided into four left and four right banks. Each bank group shares a 64-bit inter-bank datalines (iBD), consisting of upper and lower 32-bit iBD lines that connect global datapath of each bank to data I/O pins shared by all the banks. To serve a read request, multiplexers select 64-bit data from the left or right iBD lines, which is serialized and sent through the 8-bit I/O pins. The I/O data rate of DDR3 devices is 8× DRAM core clock frequency (burst length of 8) so that there are 8× more iDB lines than data I/O pins.

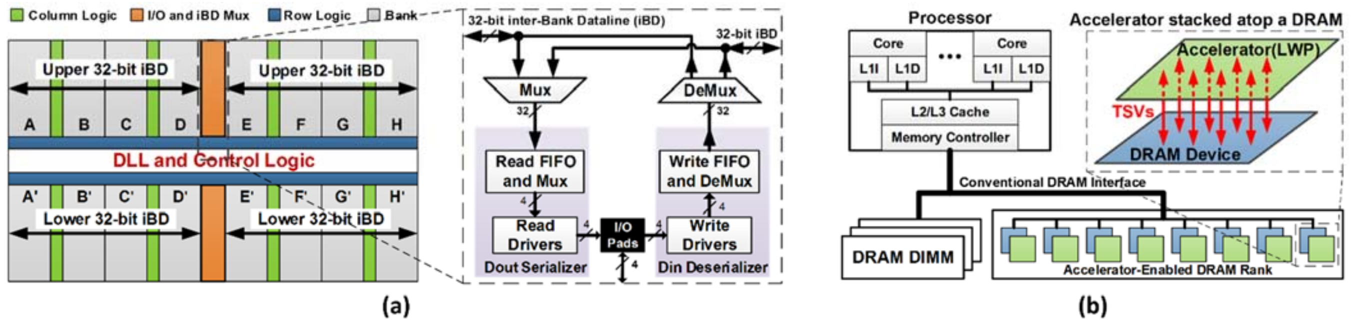

Figure 1: (a) A DDR3 device with 8 banks of A to H and ×8 I/O interface (left) and DRAM I/O datapath (right). (b) NDA architecture with light-weight processors stacked atop DRAM devices.

2.2 Single-ISA Heterogeneous Processing

Amongst the various candidates of near-DRAM acceleration architectures, such as GPU, FPGA, and coarse-grain reconfigurable accelerator (CGRA), placing processor cores with short SIMD engines and the same ISA as the host processor atop DRAM devices can be intriguing. By controlling key design parameters of CPU cores, such as issue width, reorder buffer size, and pipeline depth, a wide range of trade-offs between energy and performance is possible [6], which led to multicore design studies (e.g., [7]) and commercialization (e.g., [8, 9]). Such heterogeneous processing is initially focused on improving energy efficiency of mobile CPUs by migrating threads from big to little cores at light computing loads [8], but the other operation scenarios such as assigning threads/tasks to these heterogeneous cores depending on their performance/efficiency requirements are gaining interest as well, exemplified by Intel Xeon Phi [9].

3. Architecting NDA with Single-ISA Heterogeneous Processing

GPU, FPGA, CGRA, or low-power cores can be accelerator logic of the NDA architecture. This paper focuses on LWPs with the same ISA as the host processor due to ease of programming and maturity in development environments. High energy efficiency offered by LWPs is important because near-memory architectures have more stringent power/thermal constraints than the host processor.

The key advantage of single-ISA heterogeneous processing over other acceleration architectures is that the code designed to exploit the accelerators can be executed without any change even if the accelerators are unavailable. It is also easy to dynamically determine whether to execute certain computational kernels near DRAM (using the light-weight or little cores) or in the host processor (where relatively powerful or big cores are populated) depending on the kernel types and big/little cores' compute capability. In addition, simpler debugging and relatively stable development environments make single-ISA heterogeneous processing a compelling alternative to GPU, FPGA, and CGRA for near data processing.

NDA facilitates energy-efficient near-memory processing by stacking LWPs atop each DRAM device. A LWP is connected to the internal DRAM datalines using TSVs, as illustrated in Figure 1(b). Each LWP operates on data stored in the associated device independently of and in parallel with the LWPs on the other DRAM devices on the DIMM(s). This NDA architecture does not require changes to the processor or DIMM interface, and only minimal changes to the underlying DRAM design. Therefore, it can be used with existing systems to accelerate NDA-enhanced applications. Furthermore, this NDA architecture allows existing un-accelerated applications to run on NDA-equipped platforms without any performance penalty [6].

Traditional host processors can offload kernels to DRAM-side LWPs. Our NDA architecture transfers data through high-bandwidth and low-energy 3D interconnects (TSVs) between DRAM devices and their corresponding LWPs without the processor's intervention and processes the data using the LWPs. This minimizes data transfers through low-bandwidth and high-energy off-chip interconnects between the processor and DRAM devices. The kernel data processed by LWPs must be distributed evenly across DRAM devices to balance computations for maximal acceleration.

3.1 Connecting Accelerator with DRAM

We propose three microarchitectures to connect a LWP and a DRAM device using TSVs. Here we assume the TSV I/O energy is 4 pJ/b, which is 80% lower than the off-chip I/O energy of a DDR3 interface (Table 1). The LWP-side memory controller (MC) manages data transfers between the two dies.

NDApD (connecting TSVs to existing iBD lines): A LWP simply reads or writes data through the TSVs connected to the existing DRAM iBD lines without modifying the underlying DRAM device design. TSVs are connected to the iBD lines between iBD multiplexers and data (de)serializer (Figure 2). This microarchitecture is similar to one developed for stacking a wide-I/O DRAM device atop a processor die [10]. NDApD has no area overhead apart from inserted TSVs (Table 2). Both the host and the LWP of NDApD adopt the same data-transfer mechanism and the same peak per-device bandwidth. However, the latter consumes 51% lower energy per 8-byte read/write data transfer than the former.

NDApD×2 (connecting TSVs to doubled iBD lines): NDApD×2 is similar to NDApD except that it has twice the iBD lines and the TSVs, accordingly. This doubles the LWP-DRAM bandwidth and entails doubling local/global I/Os but reducing the number of column select lines by half per bank.

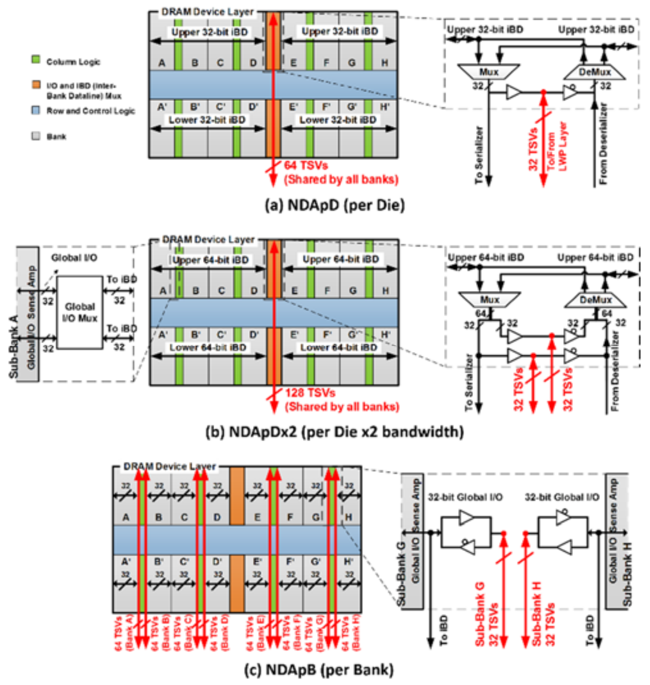

Figure 2: NDA microarchitectures.

NDApD×2 is more efficient than NDApD in read/write accesses because the overhead of sending address/command is amortized over more bits per data transaction. The LWP utilizes all iBD lines whereas the host processor uses only a half of iBD lines.

NDApB (connecting TSVs to global I/Os of each bank): All the banks in a DRAM device share the TSV connection in both NDApD and NDApDx2. In NDApB, we instead provide each independently controllable bank with a separate TSV connection to the LWP. NDApB can exploit the benefit of bank-level parallelism by accessing data from multiple banks concurrently. In this microarchitecture, we connect TSVs to global I/Os of each bank, forming eight independently accessed TSV connections. NDApB increases the LWP-DRAM bandwidth substantially with only small overhead associated with TSVs (Table 2). The latency and energy of accesses initiated from LWPs decrease as well due to iBD bypassing. The LWP-side MC directs DRAM requests/responses between the LWP and its corresponding bank.

TSV overhead: We assume coarse-grain TSVs (pitch size 50μm) for power/ground and fine-grain TSVs (pitch size 5μm) with 4:2 redundancy [11] for signals. Coarse-grain TSVs provide enough room for PHY, test, and ESD protection while fine-grain TSVs with redundancy improves the yield. With this scheme, the area overhead of NDApB that requires the most TSVs among the three microarchitectures is 0.243mm^2, which is negligible considering that the typical data area of DDR3/4 is ranging from 50 to 100 mm^2 [12].

Summary: We quantify the overheads and savings through SPICE simulations assuming 3-metal layers, 28nm DRAM process, and 6F^2 cells. Table 2 lists bandwidth, area, timing, and energy of the three microarchitectures based on a DDR3-1600 DRAM device. In all microarchitectures, the underlying DRAM architecture remains intact, making the proposals low-cost and compatible with the conventional DDR3 devices. Table 2 indicates that our microarchitectures have a limited impact on the DRAM area. DRAM access latency does not increase either; the LWP-DRAM access latency even decreases by 6ns in NDApB. LWPs consume less energy to transfer the same number of bits than host processors because LWPs obviate the need of serializing data and also the TSVs used for the LWPs have lower load capacitance than the pads, bumps, and PCBs that are used to connect a processor and DRAM packages. The wider datapath structures improves energy efficiency further because the overhead of sending/decoding a command/address is amortized over more bits per data transfer transaction.

3.2 Communication to Memory and Host by LWPs

Memory Controller: Each LWP requires an MC to issue memory requests to its DRAM device through TSVs. For all the LWP-DRAM connections proposed above, the LWP-side MC manages data transfers between the LWP and DRAM device in compliance with DRAM timing constraints including tFAW and tRRD. The complexity of the LWP-side MC can be lower than that of the host-side MC because many kernels running on LWPs include little or no data-dependent data accesses and pointer chasing.

DRAM Ownership Transition between a Host and LWPs: When a kernel is launched, the host-side MC hands over the control of a DRAM rank to LWP-side MCs. The host-side

Table 1. Area, timing, and energy parameters of an 8Gb DDR3-1600 ×8 DRAM device. ×8 = off-chip data I/O bit-width is 8.

Device	Row buffer size	Peak bandwidth	Core Freq.	I/O Freq.	Number of banks	Area	Access latency (*tCL*)	RD or WR energy without I/O	Off-chip I/O energy
DDR3-1600 ×8	1KB	12.8 Gbps	200 MHz	800 MHz	8	80 mm^2	13.75 ns	13 pJ/b	20 pJ/b

Table 2. Summary of bandwidth, area, timing, and energy of different microarchitectures to connect Accelerators and DRAM devices. Numbers are relative to DDR3-1600 ×8 parameters given in Table 1. S/A = Sense Amplifiers.

Connection between LWP and DRAM	Number of data TSVs	Peak bandwidth (Gbps)		Area change	LWP timing change	CPU timing change	LWP RD energy (without I/O)	CPU RD energy (without I/O)
		Through I/O pins	Through TSV pins					
NDApD	64	12.8	12.8	0.2% (TSV)	-	-	-51% (-7%)	-
NDApD×2	128	12.8	25.6	3.3% (S/A, TSV)	-	-	-64% (-39%)	~0% (+0.3%)
NDApB	512	12.8	102.4	1.2% (TSV)	Decrease tCL by 6ns	-	-59% (-28%)	~0% (+0.1%)

MC stops sending commands to the DRAM rank with active LWPs. This is to avoid concurrent accesses by both the host processor and the LWPs. When the ownership of a DRAM rank switches, its banks are in the precharged states. Thus, bank conflicts are avoided. When the host processor needs to access the DRAM ranks with active LWPs, LWP operations are suspended by writing to a DRAM mode register and the LWP-side MCs close (precharge) DRAM pages. At the same time, the host-side MC assumes that all DRAM row buffers in that rank are currently deactivated when it takes back the control of the DRAM rank. Then, the host-side MC can activate the required DRAM row and access data.

In a multi-programmed environment, the host processor might access the DRAM rank with active LWPs. One way to alleviate such a problem is to partition main memory space into two groups: addresses mapped to conventional and LWP-enabled ranks, respectively, guided by memory allocation policies. Memory requests from applications that do not utilize LWPs will not interfere with memory requests of active LWPs. Similarly, the same ownership transition mechanism can be used when the host-side MC must refresh the rank with active LWPs. Periodic refresh operations from the host-side MC are unavoidable, but refresh intervals are long enough (e.g., 7.8μs in DDR3) to allow LWPs to make considerable progress in the interim. LWPs are aware of refresh intervals and can suspend their normal operations right before the refresh command from the processor-side MC.

Host-LWP Communication: The host processor utilizes DRAM memory-mapped registers to communicate with LWPs through the host-DIMM interface. The host-side MC transfers necessary parameters and triggers kernel execution on LWPs by writing to the kernel registers. This avoids changing the host-DIMM interface as existing MCs support accessing programmable mode registers in conventional DRAM. LWPs can notify their execution status to the host by writing to the status registers in their local DRAM devices. The host periodically polls these memory-mapped status registers to check for kernel completion or exceptions. The status register read by the host does not interfere with DRAM accesses by the LWP because the status register that is supplied onto the off-chip bus uses a separate datapath in DRAM.

4. Software Issues in Exploiting NDA

Target Applications: Data processing in many computer vision, scientific, and engineering data-intensive applications can be split into smaller computations executed in parallel with partitioned data. Exploiting LWPs for energy-efficient, accelerated data processing and high-bandwidth and low-energy 3D interconnects, NDA can process and analyze big data effectively. The host processor can orchestrate data sharing between LWPs by duplicating some of boundary data to reduce inter-LWP communication. After LWPs complete computations, their output data is processed and merged by the host processor.

Programming Model, Cache Coherence and Memory Consistency: NDA can adopt programming and memory models similar to ones developed for heterogeneous computing CUDA and OpenCL. In such computing models the host processor is responsible for running the OS and the sequential fraction of a given application, while LWPs execute the data-intensive parallel kernels. In managing memory coherency and consistency between the host processor and the LWPs (near DRAM devices), NDA architecture also borrows current implementations developed for discrete GPUs performing explicit data transfers between the host processor and GPU memory spaces. The cost of the data transfers is typically amortized because many GPU kernels dominate the total execution time and iteratively compute on data transferred to its memory space before they transfer back the final computed result to the host processor memory space.

Supporting Virtual Memory: The host processor and the LWPs that run an application share a virtual memory space. The host processor handles the page table entries, whereas the LWPs cannot access the entries directly. When a kernel launches, the host sends active page information to the LWPs, which can then detect page faults. Page allocation is disabled in the LWPs. A LWP terminates the kernel when it experiences faults and exceptions that cannot be served by itself, such as system calls.

Arranging Data for Acceleration: a 64-bit data block is interleaved across DRAM devices in a rank when used by the host processor whereas a LWP in NDA is mapped to a single DRAM device. Because the LWP should process an entire block, not its fragment, the host processor should shuffle data before/after LWPs utilize it. This shuffling overhead is amortized as the data transfer cost is amortized in discrete GPUs.

5. Evaluation

5.1 Methodology

For evaluation we take 10 benchmarks from the San Diego Vision (DISP and SIFT) [13], Parboil (LBM and MRIG) [14], CORAL (HACC) [15], Splash-2 (OCN) [16], and Rodinia (KM, SRAD, HS, and BP) [17] suites. Table 3 summarizes the characteristics of the tested applications.

We extend gem5 with the ARM ISA to model and simulate our NDA architecture, where we consider ARM's Cortex A15- and A7-like processors for the host and light-weight processors, respectively (Table 5). We use McPAT [18] to model energy consumption of host and lightweight processor cores, caches, and other on-chip units. As detailed by our prior work [5], we integrate modified DRAMPower [19] with gem5 to evaluate energy consumption of DDR3-1600 DRAM devices and their off-chip and TSV I/O interfaces (Table 5).

5.2 Result

We compare the performance of NDA against the baseline architecture where the host processor runs a given benchmark. We run the whole application using both the host processor and LWPs while LWPs run only the kernels. In our NDA architecture, we conservatively assume that it takes 1.25ns to latch (to minimize skew) and transfer data between a DRAM device and a LWP over TSVs.

Table 3. Benchmarks used in our evaluation

Name	Description	# of Kernels	Iterative?	Replaced Exec. Time	Shuffling Overhead
BP	Back propagation [17]	4	Yes	99.9%	1.6%
DISP	Disparity map [13]	10	No	99.9%	0.1%
HACC	Hardware accelerated cosmology code [15]	2	Yes	99.9%	1.5%
HS	Hotspot [17]	2	Yes	99.9%	1.1%
KM	K-means clustering [17]	1	Yes	99.9%	0.7%
LBM	Lattice-Boltzmann method fluid dynamics [14]	1	Yes	99.9%	1.3%
MRIG	Magnetic resonance imaging gridding [14]	1	No	98.5%	0.14%
OCN	Ocean movements [16]	16	Yes	81.7%	1.6%
SIFT	Scale-invariant feature transform [13]	4	No	92.7%	0.1%
SRAD	Speckle reducing anisotropic diffusion [17]	3	Yes	99.9%	0.2%

Table 5: System configuration parameters.

Host Processor	
Core	Frequency (2GHz)
	4-way out-of-order
	ROB/IQ/LQ/SQ (128/36/48/32)
	INT/FP ALU (3/2)
Caches	L1I/L1D/L2 Size (32KB/ 32KB/512KB)
	L1I/L1D/L2 Associativity (4/4/8)
	L1I/L1D/L2 Latency (3/3/16)
LWP (1 core per DRAM die, total of 8 cores)	
Core	Frequency (1.2GHz)
	2-way in-order
	INT/FP (1/1)
Caches	L1I/L1D Size (32KB/ 32KB)
	L1I/L1D Associativity (2/4)
	L1I/L1D Latency Cycles (2/2)
Main Memory Subsystem	
Memory Controller	Page Policy (open)
	Scheduling Policy (FR-FCFS)
	RD/WR Request Queues (40/40)
DRAM (DDR3-1600, x8)	tRCD/tCL/tRP (13.75ns/13.75ns/13.75ns)
	tRAS/tCCD/tWTR (35ns/5ns/7.5ns)
	tWR/ tRTP/ tRTW (15ns/7.5ns/2.5ns)
	tRRD/tFAW/ tBURST (40ns/6.25ns/5.0ns)
	tRFC/tREFI (300ns/7.8μs)

Table 4: bandwidth utilization.

	Baseline	NDApD	NDApDx2	NDApB
DISP	15.1%	34.9%	42.6%	37.0%
HACC	6.1%	22.8%	25.1%	23.1%
HS	9.2%	22.5%	24.9%	23.4%
KM	1.6%	12.1%	12.7%	12.0%
LBM	8.7%	44.9%	46.9%	45.3%
MRIG	1.6%	5.8%	5.8%	5.8%
OCN	14.5%	40.1%	41.6%	40.5%
SIFT	3.4%	25.5%	28.6%	27.2%
BP	9.4%	7.2%	7.4%	7.3%
SRAD	8.7%	20.4%	22.2%	20.9%

Bandwidth Utilization: Table 4 compares the bandwidth utilization of the baseline with that of NDApD, NDApD2x, and NDApB. On average, our NDA architecture provides more than 3× higher bandwidth utilization than the baseline because each LWP independently accesses its corresponding DRAM device (more flexible, fine-grain DRAM accesses) whereas the baseline accesses all DRAM devices in a lock-step manner.

Speedup: The triangles in Figure 3 represent the speedup of our NDA architecture over the baseline. They show that our NDA architecture improves the performance of given applications. The speedup numbers include execution time of both software and accelerated kernel(s). High speedups originate from (1) concurrent execution by LWPs; (2) lower memory access latency for 3D-stacked LWPs atop DRAM devices; (3) each LWP's independent accesses of each DRAM device, which improves the utilization of given bandwidth compared to the baseline where all DRAM devices must be accessed in a lock-step manner (cf. Table 5); and (4) higher memory bandwidth of NDApD×2 and NDApB for LWPs, respectively. Some benchmarks such as MRIG do not take advantage of our NDA architecture because they exhibits notable temporal data locality and benefits from the host processor's large L2 cache.

NDApD, NDApD×2, and NDApB provide 1.91×, 2.06×, 1.95× higher geometric-mean performance than the baseline, respectively. We observe that NDApD×2 and NDApB do not lead to significantly higher performance than NDApD. We identify two primary reasons. First, the limited computing capability of LWPs, each of which can use at most two ALUs per cycle, is insufficient to fully exploit the higher bandwidth of NDApD×2 and NDApB. In contrast, each CGRA has a sufficient number of ALUs and thus parallel memory requests to utilize the given bandwidth of NDApD×2 and NDApB. Furthermore, NDApB with eight 64-bit independent memory channels (one channel per bank) for a LWP offers lower performance than NDApD×2 with a single 128-bite memory channel for a LWP. This is because a LWP is an in-order core and it cannot efficiently utilize the NDApB providing 8 independent 64-bit memory channels while NDApD×2 transfers more data per one 128-bit wide memory channel (more data transfer per memory access). That is, NDApD×2 offers shorter latency per 64B memory request (i.e., cache line size) than NDApB. Second, LWPs have their own local L1 caches while CGRAs do not. These local L1 caches diminish the benefit of higher bandwidth provided by NDApD×2 and NDApB.

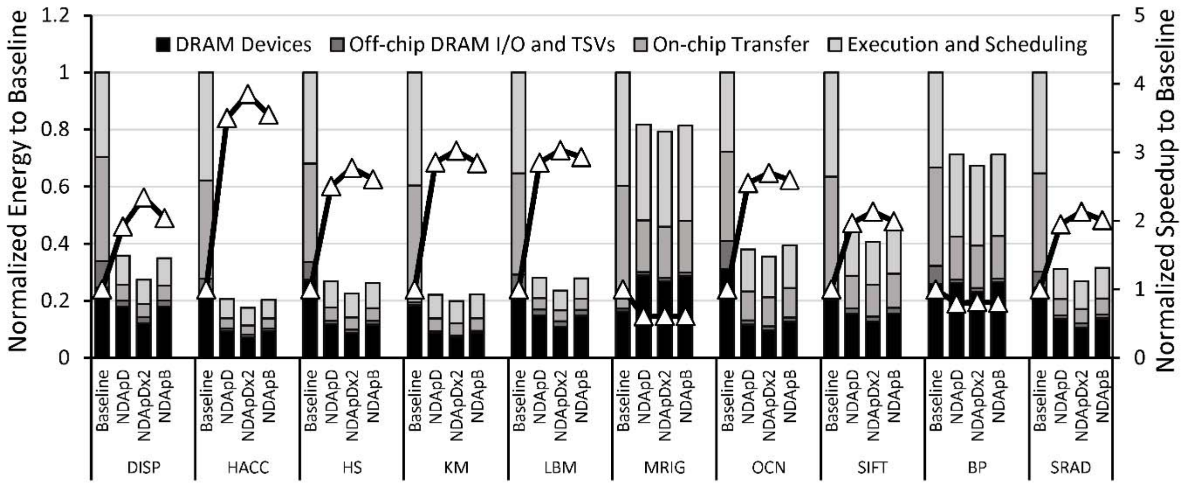

Figure 3: Comparison of energy and speedup amongst NDApD, NDApDx2, and NDApB, normalized to Baseline.

Energy: NDA improves total energy consumption by exploiting energy-efficient computations using LWPs near DRAM devices and energy-efficient data transfers using TSVs. The stacked bars in Figure 3 show the energy dissipation of the baseline and NDA architectures NDApD, NDApD×2, and NDApB consume 64%, 68%, and 64% lower geometric-mean energy than the baseline, respectively. The lower energy consumption is mainly contributed by three factors: (1) lower data transfer energy through 3D interconnects, (2) higher energy-efficiency of LWPs than the host processor; and (3) lower processor leakage and DRAM background energy due to shorter execution time. On average, the power consumption of the host processor is 5.1W while that of a LWP is 0.28W. Lastly, our NDA architecture consumes considerably lower “DRAM device” energy due to more efficient DRAM accesses (i.e., higher DRAM page (row buffer) hit rates) facilitated by each LWP’s independent access of each DRAM device.

6. Conclusion

We proposed an NDA architecture enabling accelerated data processing near main memory without changing the host processor design. NDA concurrently reduces energy and enhances throughput in data movement by stacking LWPs atop conventional DRAM devices and by off-loading data-intensive operations to the LWPs. By making the ISA of the stacked LWPs atop DRAM the same as that of the host processor, our NDA architecture is designed to be more easily adopted in both existing and emerging systems. We show that our NDA architecture improves the performance of a wide range of evaluated applications by 1.91-2.06× and reduces the energy consumption by 64-68%.

Acknowledgments

This work was supported in part by grants from NSF (CNS-1600669 and CNS-1557244) and Samsung Semiconductor. Nam Sung Kim and Jung Ho Ahn are the co-corresponding authors. Nam Sung Kim has a financial interest in Samsung Semiconductor.

References

[1] S. Keckler, et al., "GPUs and the Future of Parallel Computing," *IEEE Micro,* vol. 31, no. 5, 2011.

[2] D. Patterson, et al., "A Case for Intelligent RAM," *IEEE Micro,* vol. 17, no. 2, 1997.

[3] D. Zhang, et al., "TOP-PIM: Throughput-Oriented Programmable Processing in Memory," in *ACM HPDC*, 2014.

[4] Q. Zhu, et al., "A 3D-Stacked Logic-in-Memory Accelerator for Application-Specific Data Intensive Computing," in *IEEE 3DIC*, 2013.

[5] A. Farmahini-Farahani, et al., "NDA: Near-DRAM Acceleration Architecture Leveraging Commodity DRAM Devices and Standard Memory Modules," in *IEEE/ACM HPCA*, 2015.

[6] O. Azizi, et al., "Energy Performance Tradeoffs in Processor Architecture and Circuit Design," in *IEEE/ACM ISCA*, 2010.

[7] R. Kumar, et al., "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance," in *IEEE/ACM*, 2004.

[8] "big.LITTLE Technology," [Online]. Available: http://www.arm.com/products/processors/technologies/biglittleprocessing.php.

[9] "Intel Xeon Phi Product Family," [Online]. Available: http://www.intel.com/content/www/us/en/processors/xeon-phi-detail.html.

[10] J. 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, 2012.

[11] U. Kang, et al., "8Gb 3D DDR3 DRAM Using Through-Silicon-Via Technology," in *IEEE ISSCC*, 2009.

[12] K. Sohn, et al., "A 1.2V 30nm 3.2Gb/s/pin 4Gb DDR4 SDRAM with Dual-Error Detection and PVT-tolerant Data-fetch Scheme," in *IEEE ISSCC*, 2012.

[13] S. Kota, et al., "SD-VBS: The San Diego Vision Benchmark Suite," in *IEEE IISWC*, 2009.

[14] "IMPACT: Parboil Benchmarks," [Online]. Available: http://impact.crhc.illinois.edu.

[15] "CORAL Benchmark Codes," [Online]. Available: https://asc.llnl.gov/CORAL-benchmarks.

[16] S. Woo, et al., "The SPLASH-2 programs: Characterization and methodological considerations," in *ISCA*, 1995.

[17] S. Che, et al., "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *IEEE IISWC*, 2009.

[18] [Online]. Available: http://www.hpl.hp.com/research/mcpat.

[19] "DRAMPower," [Online]. Available: http://www.es.ele.tue.nl/drampower.

Biographies

Hadi Asghari-Moghaddam is a PhD student in the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. He received the BS degree in electrical engineering in Sharif University of Technology and the MS degree in electrical and computer engineering from the University of Wisconsin-Madison. His research interests include near-data computing, and energy-efficient computer architecture. Contact him at asghari2@illinois.edu

Amin Farmahini-Farahani is a member of the technical staff at Advanced Micro Devices, where he works on research projects on processing in memory. His research interests include memory systems, energy-efficient processing, and reconfigurable computing. He has a PhD in electrical and computer engineering from University of Wisconsin-Madison. Contact him at afarmahi@amd.com.

Katherine Morrow is an Associate Professor in the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison. She received her BS, MS, and PhD from Northwestern University. She is a member of ACM and IEEE. Her research interests include myriad aspects of reconfigurable computing, and her teaching interests focus on the use of technology to improve student learning in undergraduate computer engineering courses. Contact her at klmorrow@wisc.edu

Jung Ho Ahn is an associate professor in the Graduate School of Convergence Science and Technology at Seoul National University, where he leads the Scalable Computer Architecture Laboratory. He is interested in bridging the gap between the performance demand of emerging applications and the performance potential of modern and future massively parallel systems. Ahn has a PhD in electrical engineering from Stanford University, and is a senior member of IEEE. Contact him at gajh@snu.ac.kr.

Nam Sung Kim is an Associate Professor in the Department of Electrical and Computer Engineering at the University of Illinois at Urbana–Champaign. His research focuses on devices, circuits, and architectures for energy-efficient computing. Kim has a PhD in computer science and engineering from the University of Michigan. He is a senior member of IEEE. Contact him at nskim@illinois.edu.