# Chameleon: Versatile and Practical Near-DRAM Acceleration Architecture for Large Memory Systems

Hadi Asghari-Moghaddam† Young Hoon Son‡ Jung Ho Ahn‡ Nam Sung Kim†
†University of Illinois at Urbana-Champaign ‡Seoul National University
{asghari2, nskim}@illinois.edu {yhson96, gajh}@snu.ac.kr

***Abstract*—The performance of computer systems is often limited by the bandwidth of their memory channels, but further increasing the bandwidth is challenging under the stringent pin and power constraints of packages. To further increase performance under these constraints, various near-DRAM acceleration (NDA) architectures, which tightly integrate accelerators with DRAM devices using 3D/2.5D-stacking technology, have been proposed. However, they have not prevailed yet because they often rely on expensive HBM/HMC-like DRAM devices which also suffer from limited capacity, whereas the scalability of memory capacity is critical for some computing segments such as servers. In this paper, we first demonstrate that data buffers in a load-reduced DIMM (LRDIMM), which is originally developed to support large memory systems for servers, are supreme places to integrate near-DRAM accelerators. Second, we propose *Chameleon*, an NDA architecture that can be realized without relying on 3D/2.5D-stacking technology and seamlessly integrated with large memory systems for servers. Third, we explore three microarchitectures that abate constraints imposed by taking LRDIMM architecture for NDA. Our experiment demonstrates that a *Chameleon*-based system can offer 2.13× higher geo-mean performance while consuming 34% lower geo-mean data transfer energy than a system that integrates the same accelerator logic within the processor.**

## I. Introduction

More applications demand higher bandwidth between the off-chip main memory and the processor for greater performance. Under the tight pin and power constraints of packages, however, the bandwidth has been stagnant because a higher data transfer rate per pin comes at the cost of worsened signaling integrity and superlinearly increased power consumption. Furthermore, interconnect capacitance has scaled at a much slower rate than logic capacitance with technology scaling [25], significantly increasing the fraction of data transfer energy in the total system energy [10], [29].

These aforementioned challenges motivate researchers to reconsider the past processing-in-memory (PIM) architectures [15], [17], [19], [28], [36], [39], [44], [47] that aimed to improve performance by integrating processor logic and DRAM on the same die. More specifically, researchers have proposed near-DRAM acceleration (NDA) architectures wherein an accelerator logic layer is 3D/2.5D-stacked atop a (custom or commodity) DRAM layer to reap the performance and energy-efficiency benefits of both accelerators and near-memory processing [20], [51], [63], [65].

978-1-5090-3508-3/16/$31.00 © 2016 IEEE

Nonetheless, the 3D/2.5D integration itself cannot automatically offer significantly higher bandwidth for accelerators than standard DRAM devices. This is because the bandwidth of each DRAM bank is often designed to match that of device I/O. To provide notably higher bandwidth for the accelerators, considerable customization of DRAM is required and it significantly increases the cost. For example, analyzing the die photo of SK Hynix high-bandwidth memory (HBM) [33], we discover that the through silicon vias (TSVs) for 1024-bit I/O consume nearly 20% of each DRAM layer, let alone a separate logic layer for PHYs. Moreover, the 3D-integration technology does not yield notably lower latency for 3D-stacked accelerators than standard DRAM devices either, because the latency is dominated not by off-chip interconnects but by DRAM cores [11], [12].

Memory systems need to serve both NDA and traditional applications (and thus support large capacity). However, popular memory substrates for NDA such as HBM and hybrid memory cube (HMC [48]) suffer from limited capacity; since they employ point-to-point connections to memory controllers for high-speed data transfers, increasing the memory capacity requires more memory controllers, channels, and I/O circuits in contrast to traditional DDR memory modules. This in turn requires (1) use of separate traditional DDR DRAM modules and (2) data transfers between HBM/HMC and DDR DRAM modules. Furthermore, HMC with an abstracted memory interface and a network of HMC modules [30] can increase the memory capacity, but it significantly increases memory access latency. Lastly, HMC uses serial links that also increase the latency of memory accesses. The serial links have high idle power consumption and will suffer from high wake-up time if they are powered-off during idle periods [50]. These aspects of HMC are not desirable for datacenter servers for on-line data-intensive (OLDI) applications. In contrast, the scalability of memory capacity and low latency of memory accesses to large memory systems are critical for big-memory servers [52].

In pursuit of more practical and less expensive NDA supporting larger memory capacity, we first notice unique aspects of DDR4 LRDIMM (load-reduced dual-inline memory module) architecture [5]. That is, a DDR4 LRDIMM is comprised of (1) a repeater (called registering clock driver (RCD)) device for command/address (C/A) signals from a memory controller to all of its DRAM devices and (2) data buffer (DB) devices for data signals (one per group of vertically aligned

standard DDR4 DRAM chips). LRDIMMs are commonly used to provide up to 8× more main memory capacity than unbuffered DIMMs (UDIMMs) without sacrificing the maximum bandwidth of commodity DDR4 devices; UDIMMs require a trade-off between capacity and bandwidth (e.g., 1600, 1866, and 2133MT/s for 3, 2, and 1 DIMMs per channel) due to signal integrity challenges [26].

The DRAM industry often seeks a solution that does not require significant customization of DRAM devices, but optionally increases DRAM values depending on the market demands. LRDIMM is one of such solutions since it can optionally increase memory capacity without customization of standard commodity DRAM devices by integrating DB devices on a DIMM.

Discovering that these DB devices can be an attractive platform to integrate accelerators near DRAM devices, we propose *Chameleon* – an NDA architecture that is built on the current DDR4 LRDIMM architecture and its standard interface. *Chameleon* can cost-effectively adapt itself for different accelerator architectures depending on target application domains since it only needs to attach different NDA-enabled DB devices to an LRDIMM board, which is much cheaper than 3D/2.5D-stacking different accelerator dies atop HBM/HMC-like memory substrates. Moreover, *Chameleon* by design can support far larger memory capacity than NDA architectures relying on HMC/HBM-like DRAM since it is built on LRDIMM.

At the same time, we also face some challenges of using DDR4 LRDIMMs for NDA. One challenge is that DB devices do not comprise dedicated C/A pins since only the RCD device can drive the C/A signals in an LRDIMM. Tackling this challenge, we propose *Chameleon-t* and *-s* – two microarchitectures that temporally and spatially multiplex data (also known as DQ) pins of DB devices not only to send/receive DQ signals but also to drive C/A signals. This new feature is a very small change in DRAM I/O part and thus easily accommodated since it does not change either the existing I/O pins and their layout of packages or DRAM bank architectures; DRAM manufacturers have accommodated major customers' requests of adding small features to mobile DRAM products. Besides, they do not affect the normal operations between the host processor and DIMMs when the NDA mode is unused. Lastly, we demonstrate that they can offer higher performance and energy efficiency than seemingly a more straightforward microarchitecture that transfers/drives the C/A signals from an accelerator in a DB device through the RCD chip.

*Chameleon-t* and *-s* may suffer from the overhead of using DQ pins for C/A/DQ signals between DB and DRAM devices. However, we demonstrate that the overhead can be compensated by increasing the data transfer rate per DQ pin (denoted by "gear-up" mode) because we disconnect the DB devices in an LRDIMM from the host memory controller and other LRDIMMs in NDA mode. Note that there are two types of connections: (1) a short single-drop connection between a DB and its coupled DRAM devices on an LRDIMM; and (2) a long multi-drop connection amongst the host memory controller and multiple LRDIMMs on the motherboard. Due to a better signaling condition, the short connection within an LRDIMM can achieve a higher transfer rate than the long multi-drop connection (e.g., GDDR5 only for a single-drop connection up to 7Gbps [8] versus DDR4 up to 2.4Gbps [43]). In ordinary operating mode, however, the long multi-drop connection limits the transfer rate between the host memory controller and an LRDIMM.

Lastly, increasing the number of DIMMs per channel does not increase the bandwidth between the host processor and the DIMMs in traditional memory systems. However, we demonstrate that *Chameleon* can scale the aggregate bandwidth exposed to accelerators by simply plugging more *Chameleon*-enabled LRDIMMs. This is because accelerators in each DIMM do not use the shared channel but the local/private channels, which connect DB devices and their coupled DRAM devices, are isolated from the shared channel by the DB devices in NDA mode.

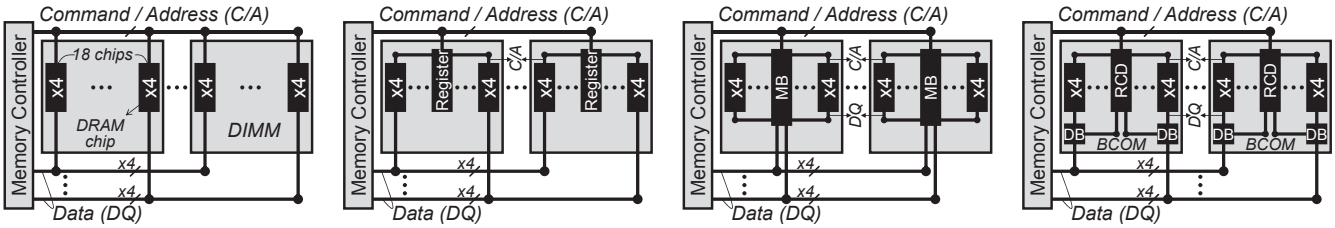
Throughout experiment, we show that *Chameleon-s* can offer 96% geo-mean performance of an NDA architecture that utilizes the same bandwidth as commodity DRAM devices but requires 3D-stacking technology to couple an accelerator with a DRAM device (i.e., "NDA-1" in [20]) over a set of evaluated parallel benchmarks. Furthermore, with one, two, and three LRDIMMs per memory channel, *Chameleon-s* can offer 14%, 74%, and 113% higher geo-mean performance than the baseline with the same number of accelerators integrated on the same chip as the host processor. In summary, we make the following key contributions:

- We propose *Chameleon*, an NDA architecture built on LRDIMM architecture after demonstrating that LRDIMM data buffer chips can be an appealing platform to position near-DRAM accelerators.
- We explore two *Chameleon* microarchitectures to tackle the key challenge of using LRDIMMs for NDA – lack of dedicated C/A pins in DB devices.
- We distinguish three key benefits of our *Chameleon* NDA architecture: compared with accelerators within the host processor on the same chip or package: (1) supporting gear-up mode for higher data rate per pin; (2) scaling bandwidth for accelerators with more LRDIMMs; and (3) utilizing given bandwidth more efficiently.

## II. Background

### A. Buffered Dual-Inline Memory Modules

In order to achieve a balance between capacity and bandwidth, multiple DRAM devices that operate in tandem compose a rank, and one or more ranks are packaged in a memory module. A popular package called Dual-Inline Memory Module (DIMM) has 64 data (DQ) pins excluding the ones for ECC (Figure 1(a)). Modern DRAM devices operate at GHz ranges and a memory controller has to drive even more than 100 of these devices to deliver command/address information through noisy channels, leading to a serious signal integrity problem. For example, a command/address (C/A) pin from a memory controller has to drive 144 DRAM devices (18




**(a)** UDIMM **(b)** RDIMM **(c)** DDR3-LRDIMM **(d)** DDR4-LRDIMM

**Fig. 1: Popular memory module types: (a) (U)DIMM, (b) RDIMM, (c) LRDIMM (DDR3), and (d) LRDIMM (DDR4). MB stands for memory buffer, RCD for registering clock driver, DB for data buffer, and BCOM for side-channel bus.**

| | DIMM | RDIMM | LRDIMM |
|---|---|---|---|
| Maximum ranks per channel | 2 | 4 | 8 |
| Additional repeater latency | N/A | 0.8ns | 2.0ns |
| Repeating C/A signals? | No | Yes | Yes |
| Repeating DQ signals? | No | No | Yes |
| Per DRAM data buffer? | N/A | No | Yes |

**TABLE I: Comparing (U)DIMM, RDIMM, and LRDIMM for DDR4-2400 [4], [5].**

$\times$4 devices per rank supporting ECC multiplied by 8 ranks) when 8 ranks are populated per channel, whereas a DQ pin is connected to 8 DRAM devices, which is an order of magnitude smaller. Therefore, modules for servers, which have larger memory capacity demands than personal computers, typically employ a register per DIMM to reduce this huge capacitive load of a memory controller, alleviating this signal integrity issue. This type of a module is called a registered-DIMM (RDIMM), as shown in Figure 1(b).

The register in an RDIMM substantially alleviates the high load of C/A pins, improving bandwidth and capacity scaling of memory modules. However, starting from DDR3 that delivers data over one giga bits per second per pin, the signal integrity of DQ buses has become a serious concern as well. A new module type called Load-Reduced DIMM (LRDIMM) [26] was introduced to improve the signal quality of DQ signals as well as C/A signals (Figure 1(c)). The heart of this LRDIMM is a memory buffer (MB) device that repeats all these signals. More recent DDR4 standard [4] also employs LRDIMM. However, the single MB device is divided into multiple pieces: a registering clock driver (RCD), one per module to repeat C/A signals and data buffer (DB) devices to further improve the signal integrity of DQ signals under tight skew and jitter constraints (Figure 1(d)) [5]; a DB device is attached to a set of DRAM devices, which are vertically placed and share a portion of DQ signals. These separate DB devices minimize a serious drawback of DDR3's memory buffer, huge spatial disparity in DQ traces amongst DRAM devices to the central buffer, and hence facilitate further improving data transfer rates. The main function of a side channel called BCOM is to transfer buffer control commands and status information between RCD and DB devices mainly during initialization. Table I compares aforementioned memory module types.

### *B. DRAM Device Microarchitecture*

A DDR DRAM device includes multiple two-dimensional banks that are tied together to constitute a DRAM core. This DRAM core communicates with a corresponding external memory controller through DRAM's I/O part including DQ and C/A pins. All the banks in a single device receive C/A information from a single set of C/A pins. Each bank exposes a single row buffer to a memory controller. In order to access data at a specific location of the bank, an entire row that includes the data should first be activated into the row buffer by an *activate* command. After accessing the data in the row buffer through a *read* or *write* command, the row should be deactivated by a *precharge* command before data at any different row in the bank are accessed. Data stored at each cell of a bank should be refreshed periodically because the charges in a cell capacitor, whose amount determines the value stored in the cell, leaks over time. The interval between consecutive *refresh* commands is around 10$\mu$s, and each command refreshes cells in multiple rows in a bank to refresh entire cells in the bank within cell retention time. As of 2016, 4Gb or 8Gb is the most popular capacity per DRAM device, each has 8 or 16 banks, and the row size is around 8Kb.

### *C. Programming/Memory Models for NDA*

NDA can adopt programming/memory models similar to ones developed for heterogeneous computing such as CUDA [38] and OpenCL [53]. In such models, the CPU (or host processor) is responsible for running the OS and the sequential fraction of a given application, while accelerator executes the compute- and/or data-intensive parallel fraction (i.e., kernels). In managing memory coherency and consistency between the host processor and accelerators (near DRAM devices), NDA architecture can also borrow current implementations developed for heterogeneous computing. For example, we can adopt a memory model used by AMD's early generation APUs providing a single physical memory space but two separate/isolated logical memory spaces for CPU and GPU, respectively; a recently proposed NDA architecture [20] and our Chameleon NDA architecture assume a similar memory model. Although such a memory model requires explicit data transfers between CPU and GPU memory spaces, the

cost of the data transfers between the CPU and the GPU is typically amortized. This is 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 CPU memory space. Otherwise, we can adapt the unified or managed memory in NVIDIA's CUDA 6 [32] and AMD's hUMA [58] where the memory model is simplified by eliminating the need for transferring data between two separate memory spaces.

### D. NDA Architecture Built on Conventional DIMMs

A recent proposal takes a conventional DIMM as a platform for NDA [20], where one or more accelerators are 3D-stacked atop each coupled DRAM device in a DIMM. In such an NDA architecture, a computation for a large input data set is split/partitioned into computations for small input data sets for accelerators; many big-memory/data-intensive applications based on a distributed computing model such as MapReduce can be split in this way [18]. In this paper, we focus on the DRAM microarchitectures exploiting DB devices, building on the prior NDA architecture [20].

The traditional DIMM's data placement policy distributes each 64-bit word across all the DRAM devices in a DIMM. Therefore, by carefully considering the host processor's DRAM address interleaving/mapping scheme and leveraging modern processor's powerful bit shuffling instructions, a *cudaMemcpy*-like API function should shuffle the input data such that each accelerator can access complete 64-bit words. In such a shuffled data placement scheme, some boundary data may need to be duplicated to minimize inter-accelerator communications. The host processor can orchestrate data sharing amongst accelerators by reading/writing data from/to DRAM devices. After computations are completed by the accelerators, the output data is unshuffled/merged by the cudaMemcpy-like API function running on the host processor.

Under such a bit shuffling scheme, a virtualized ECC scheme [61] can protect the memory space for each accelerator if ever needed, whereas the conventional rank-level ECC can still protect the host memory space. The host processor still uses page-based virtual memory for its memory space. In contrast, the accelerators access their memory space using physical memory addresses as GPUs in early AMD's APUs do. That is, we adopt memory segmentation that maps contiguous regions of virtual memory to contiguous regions of physical memory as we can eliminate the need for virtual memory management and target big-memory/data-intensive applications that rarely benefit from paging [9]. Lastly, the details of accelerators managing DRAM devices (e.g., memory requests and refresh commands) and DRAM ownership transitions/communications between the host processor and accelerators are described in [20].

## III. Chameleon NDA Architecture

As described in Section II-A, a DDR4 LRDIMM is comprised of DRAM, RCD, and DB devices (Figure 1(d)); as the signaling rate of DDR4 and future DDR is significantly higher

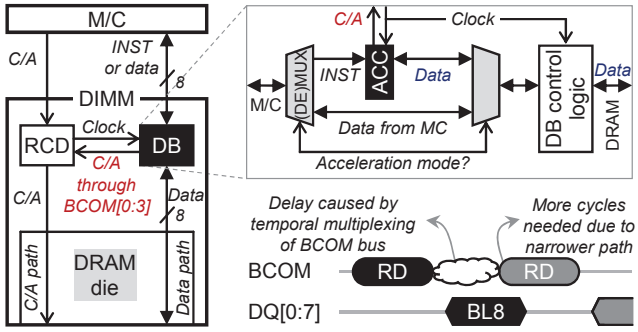



**Fig. 2: A straightforward NDA microarchitecture leveraging DDR4 LRDIMM architecture called *Chameleon-d*. INST stands for instructions from memory controllers heading to Accelerators (ACC), C/A for command/address, and BL8 for burst length of 8.**

than that of DDR3, it is inevitable to split an MB device used for DDR3 LRDIMM into RCD and DB devices in DDR4 LRDIMM under the tight form-factor constraint of DIMM (cf. Section II-A). This in turn poses unique challenges for NDA compared with a DDR3 LRDIMM, where an accelerator integrated onto the single MB device can transfer data from/to DRAM devices as the host processor does through its memory controller. First, an accelerator integrated onto each DB device of a DDR4 LRDIMM can access only a 4- or 8-bit fraction of each 64-bit word. Second, the accelerator in each DB device cannot dispatch its own C/A signals to its vertically coupled DRAM device. This is because each DB device has pins for only data signals (DQ) with a few more pins for a side-band channel (BCOM) connecting to the RCD device. While the first challenge can be overcome by adopting a prior technique of shuffling and partitioning input data [20], the second challenge is an unchartered one yet to be tackled in this paper.

To obviate the two aforementioned issues, we may consider adding dedicated DQ buses between DB and RCD devices such that accelerators in the RCD device perform NDA. However, the PCB of a DIMM that houses DRAM, RCD, and DB chips has limited signal routing space. Thus, creating an additional interface transferring data between DB and RCD is practically impossible under signal integrity and other constraints; splitting the MB device in DDR3 (which buffers both C/A and DQ signals) into DB and RCD in DDR4 was the very same reason. Furthermore, the limited RCD command bandwidth will become the performance bottleneck (see the evaluation associated with the baseline *Chameleon-d* in Section VI-A).

Figure 2 depicts a seemingly straightforward NDA microarchitecture leveraging LRDIMM architecture and its standard interface. This NDA microarchitecture decouples the function of generating and transmitting C/A signals from DB devices by leveraging the traditional function of an RCD device (i.e., repeating C/A signals from a memory controller for connected DRAM devices in an LRDIMM). In an LRDIMM,

an RCD device utilizes BCOM bus/pins to exchange control and status information that originates from and heads to the corresponding host memory controller. We can modify this BCOM interface design such that once the accelerator in a DB device becomes a master, the BCOM bus delivers C/A signals to access the coupled DRAM devices to the connected RCD device. This allows us to dedicate the full bandwidth of DQ pins/bus only for data transfers and to use conventional DRAM devices with no modification. However, its seeming advantage comes at the cost of limited C/A bandwidth between the accelerators and the coupled DRAM devices. More specifically, all the DRAM devices within a module are designed to receive the same C/A signals. Therefore, time multiplexing or arbitration is inevitable to transfer C/A signals generated from multiple accelerators in this microarchitecture, which may significantly hamper overall performance. For the purpose of comparison with our two proposed NDA microarchitecture in the subsequent sections, we denote this NDA microarchitecture by *Chameleon-d*.

Figure 3 depicts *Chameleon* NDA architecture with spatial and temporal microarchitectures denoted by *Chameleon-s* and *-t* to tackle the aforementioned second challenge. Both microarchitectures have a pair of (de)multiplexers in a DB device for providing a path to an accelerator. Thus, when the NDA mode is unused, the main memory system operates as traditional LRDIMMs with negligible impact on timing; compared to the latency of lengthy wires that span an entire die and a series of sense amplifiers in the device, that of a single-stage (de)multiplexers is a tiny portion of a DRAM clock cycle (tCK). More specifically, *Chameleon-s* and *-t* utilize the existing bidirectional DQ pins not only to transfer data between a DB device and its coupled DRAM devices but also to dispatch C/A pairs from the DB device to the coupled DRAM devices.

Lastly, *Chameleon* NDA is also applicable to DIMMs with LPDDR and GDDR that support 32-bit I/O per chip and obviate the need for data shuffling as each device can store complete 32-bit words [49]; there were proposals using LPDDR chips with buffer chips for low-power data center DIMMs [40], [60].

### A. Chameleon-t: Temporally Multiplexing DQ Bus

*Chameleon-t* exploits time multiplexing in utilizing the DQ pins/bus for not only transferring data between a DB device and its coupled DRAM devices but also dispatching C/A pairs from the DB device to one of the coupled DRAM devices (Figure 3(a)). Suppose that we use 16Gb $\times$8 DRAM devices. Each DRAM device has a burst length of 8 (= 4 tCK with a double data rate (DDR)). Because C/A signals are transmitted in a single data rate (SDR) within DDR4 devices, a C/A pair would occupy 1 tCK and 2 tCK, where up to 16 and 32 bits can be included, respectively. The JEDEC standard [4] defines the page size of this device type as 8Kb and the number of banks as 16. Therefore, an *activate* command requires $log_2(^{16Gb}/_{8Kb}) = 21$ address bits. Assuming that a command can be encoded with 3 bits, 2 tCK should be needed to send an activate command. The number of address bits for a read or write command is $log_2(16\ banks) \times (^{8Kb}/_{burst8\times8}) = 11$, which can be packed in 1 tCK. A *precharge* command only accompanies bank information, where 1 tCK is enough. Considering the burst length of 4 tCK, we pack a series of commands (e.g., activate, read, and precharge commands to read a DRAM burst ($(1+2+1=)$ 4 tCK)) in one burst.

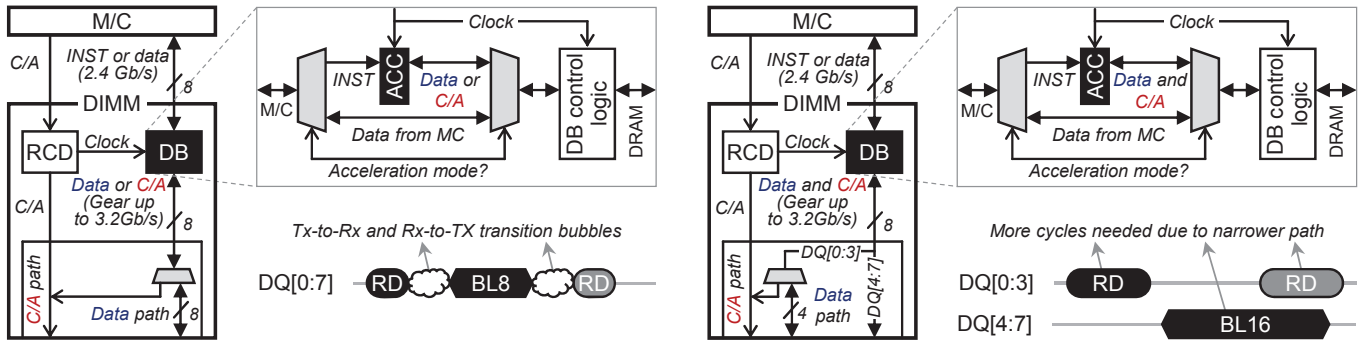
Spatial locality in memory accesses leads to multiple column accesses per activated row (lower row-buffer conflict rates [31]), alleviating this burden to the DQ bus, but it is still significant. One way to further reduce this overhead is to combine an activate command with its subsequent read or write command. In this case, the number of address bits is $log_2(^{16Gb}/_{8b\times8}) = 28$, which can be packed in 2 tCK even if command signals are included. Read and write with auto-precharge commands can also reduce DQ bus occupancy by command transaction.

Minimal changes are needed at the I/O part of a DRAM device to support *Chameleon-t*. In NDA mode, a DRAM device treats signals from the coupled DB device as C/A signals by default. The only exception is a predetermined cycle after that it receives a *write* command, when the demultiplexer that is located between DQ pins and the remainder of the DRAM device should forward the signals from the DQ pins to the inter-bank datalines. An aligner is needed between the demultiplexer and the remaining C/A path because the number of DQ pins is different from that of C/A pins. Note that sharing DQ pins with C/A pins was implemented in the previous DRAM interfaces such as FBDIMM [3], [21] (between a memory controller and the advanced memory buffers in FBDIMM, where C/A/DQ signals are (de)multiplexed to communicate with DRAM chips) and its enhancement [27]; this proposed change is confined to the device I/O parts and thus it can be easily implemented in the current DDR4 devices at a negligible cost according to industry experts.

### B. Chameleon-s: Spatially Multiplexing DQ Bus

In contrast to the time multiplexing of DQ pins/bus in *Chameleon-t*, *Chameleon-s* splits the bidirectional DQ pins/bus in space and dedicates a few lanes of the bus/pins for C/A signals (Figure 3(b)); we consider the number of pins dedicated for C/A signals as a microarchitecture design parameter and conduct experiments in Section VI-A. Fixing the number of lanes for C/A signals leads to inefficient usage of these lanes because the number of commands per data transfer varies depending on memory request patterns. Because fewer DQ pins are used for data signals in NDA mode, more cycles are needed to transfer the same number of bytes, which effectively increases the burst size. In Figure 3(b), 4 out of 8 pins are used for data transfers, increasing the burst length from 8 to 16 in NDA mode. Similar to *Chameleon-t*, aligners are needed for both C/A and DQ signals within a DRAM device, but the demultiplexer is not necessary.

Such a spatial separation of DQ pins for data and C/A signals reduces the frequency (overhead) of signaling direction changes. The main memory is more frequently read than




**(a)** *Chameleon-t* **(b)** *Chameleon-s*

**Fig. 3: Two *Chameleon* NDA microarchitectures with (a) temporal (*Chameleon-t*) and (b) spatial (*Chameleon-s*) C/A/DQ signals. INST stands for instructions from memory controllers heading to Accelerators (ACC), C/A for command/address, and BL8 and BL16 for burst length of 8 and 16, respectively.**

written while *Chameleon-t* experiences at least one direction change per read (a read command from an accelerator and a read data from a DRAM device), whereas *Chameleon-s* has no direction change for consecutive memory reads or writes. Even if the physical distance between a DB device and its coupled DRAM devices is short (up to an inch), this bus ownership change incurs a bubble in time during which no device in the bus can send any signal.

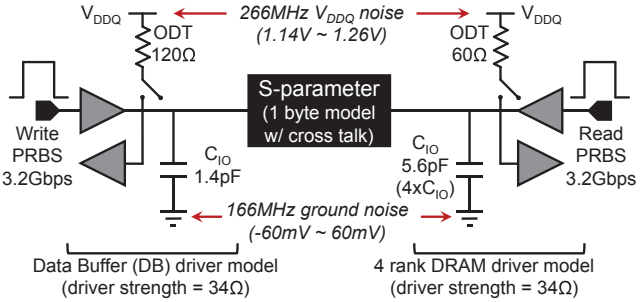
## IV. Transcending the Limits of Standard (LR)DIMMs

*Chameleon* NDA architecture that shares the DQ pins for both C/A and DQ signals inherently suffers from lower bandwidth and thus offers lower performance than the prior NDA architecture that 3D-stacks accelerators atop commodity DRAM devices of DIMMs. Furthermore, NDA architectures built on DIMMs (e.g., *Chameleon* and [20]) do not advocate any significant change in commodity DRAM core and DIMM architectures. Thus, the aggregate bandwidth between accelerators and their coupled DRAM devices in NDA DIMMs seems the same as the bandwidth between accelerators within the host and DRAM devices in standard DIMMs.

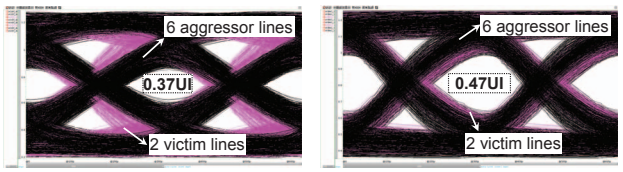
In this section, we first propose a gear-up mode that can make *Chameleon-t* and *-s* as competitive as the prior NDA architecture. Second, we deliberate how the NDA architectures built on DIMMs can offer higher performance than the traditional architecture, highlighting two aspects.

### A. Higher Transfer Rate per DQ Pin – Gear-up Mode

In an LRDIMM there is a local channel between DB and DRAM devices. As illustrated in Figure 5, DB devices in an LRDIMM are disconnected from the global memory channel shared with other LRDIMMs in NDA mode, which makes the local channel private to the accelerators. Subsequently, they directly control and communicate with their coupled DRAM devices through the local/private channel. The local/private channel between DB and DRAM devices is one inch or less in distance and has almost symmetric wiring traces. In contrast,




**(a)** SPICE model




**(b)** TX eye diagram **(c)** RX eye diagram

**Fig. 4: (a) SPICE simulation model, (b) DB-to-DRAM (TX) result, and (c) DRAM-to-DB (RX) result to check if per-DQ data transfer rate between a DB and DRAM devices can achieve 3.2Gb/s when 4 DRAM devices are connected to a DB device.**

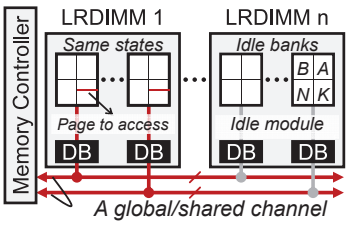
the global/shared channel between a memory controller and DB devices can be several inches in distance and it has multiple stubs that distort traversing signals, all of which limit the data transfer rate of standard DDR4 channels. Therefore, the local/private channel offers a much better signaling condition for *Chameleon-t* and *-s* than the global/shared channel, and the data transfer rate of the local/private channel can surpass that of the global/shared channel, which is assumed to be 2.4Gb/s in this paper.

We utilize the existing clock delivery network between an RCD, DRAM, and DB devices as shown in Figure 3. Adjusting the clock rate only affects the signaling rates of I/O pins and does not change the latency and throughput

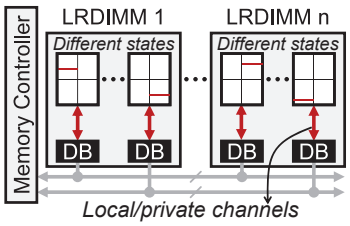
of DRAM cores. Low-power DDR DRAM devices, such as LPDDR4 [6], already support a variable I/O clock rate. We call this feature gear-up, in contrast to the gear-down mode in the DDR4 standard, which was introduced to alleviate the signal integrity issue of C/A signals by reducing their data transfer rate into half when per-DQ data transfer rate is 2.4Gbps or higher [4]. Although it takes a non-negligible amount of time to change the I/O clock rate of DRAM devices (in the order of a $\mu$s [6]), this gear-up mode is enabled only in NDA mode. That is, the cost of changing I/O clock rates is amortized since the ownership of DRAM devices does not frequently switch between DB devices and the memory controller as deduced from Section II-C.

SPICE simulation results demonstrate the feasibility of the gear-up mode. We setup a system that connects a DB device to four DRAM devices, which is the maximum number supported by DDR4 LRDIMM. We modeled the parasitic components of the interconnect between the devices by using Micron's 8-port S-parameter model that also considers the crosstalk between interconnects [42] as shown in Figure 4(a). We used pseudo random bit sequence (PRBS7) that generates data at the rate of 3.2Gbps per pin. The DRAM I/O model, which consists of transmitter/receiver models and load capacitance ($C_{IO}$ (= 1.4pF per receiver)), is based on the DDR4 SPICE model provided by Samsung [2]. The strength value of drivers in the DB's physical interface and on-die termination (ODT) values follow the default specification in the JEDEC standard [4], where ODT values are 120Ω and 60Ω for read and write mode, respectively, for the tested configuration. We also modeled power and ground noise induced by DRAM operations.

Figure 4 shows the eye diagram of data transfers from the DRAM to DB device (Figure 4(b)) and from the DB to DRAM device (Figure 4(c)). At the data transfer rate of 3.2Gbps, we observed that the receiver timing width is 0.37UI for the DRAM-to-DB path and 0.47UI for the DB-to-DRAM one, respectively, when we set the receiver mask voltage to the value specified by the JEDEC standard (= 0.136V). UI is the duration of a single-bit data transfer through a DQ pin, which is equal to 0.5 tCK. Considering that the minimum receiver timing width specified by JEDEC is 0.2UI, we secured sufficient eye size and conclude that increasing the per-DQ data rate from 2.4Gb/s to 3.2Gb/s is feasible for both *Chameleon-t* and *-s*. As we further increased the data transfer rate to 3.733Gb/s, the receiver timing width became smaller than 0.2UI, showing that 3.2Gb/s is the upper-bound of the gear-up mode. Note that there are currently DDR4 products that achieve 3.2Gb/s or even higher per-DQ-pin data transfer rates. However, this overclocking is achieved through increasing the operating voltage (e.g., 1.4V instead of 1.2V) and exploiting timing margins of DRAM devices, an approach similar to [34]. Our gear-up mode is different from the overclocking in that it increases the data transfer rate between DRAM devices and the corresponding DB device not by increasing the operating voltage but by exploiting better signaling condition of the local/private channel.




**(a)** ACCinCPU




**(b)** *Chameleon*

**Fig. 5: (a) A conventional channel has fixed data bandwidth regardless of the number of LRDIMMs populated, whereas (b) the aggregate bandwidth to the accelerators in *Chameleon* is proportional to the number of LRDIMMs in NDA mode.**

### *B. Higher Aggregate Bandwidth and Better Bandwidth Utilization Enabled by Chameleon*

Increasing the number of DIMMs per channel does not increase the bandwidth between the host processor and the DIMMs because the channel is shared by all the DIMMs in a traditional main memory system (Figure 5(a)). In *Chameleon* NDA architecture, however, accelerators in each LRDIMM do not use the shared channel but use the private channels that connect DB devices and their coupled DRAM devices and are isolated from the shared channel by the DB devices in NDA mode, which was also pointed out by [49]. Consequently, the aggregate bandwidth, which can be utilized by the accelerators, is multiplied by the number of LRDIMMs per channel as depicted in Figure 5(b). In other words, the bandwidth per accelerator remains constant in NDA architecture while it is divided by the number of accelerators in traditional architecture that integrates the accelerators within the host processor itself.

In a DDR3 LRDIMM the MB device can observe and control all the C/A and DQ signals. Suppose a hypothetical NDA architecture using a DDR3 LRDIMM where accelerators are in its MB device. Such an NDA architecture is straightforward and compatible with the traditional DIMM's operations (i.e., transferring 64 bytes over 8 bursts per memory request) since it does not need to shuffle and partition input data. However, many researchers have pointed out inefficient bandwidth utilization of traditional DIMM architecture by various applications, also known as a DRAM overfetching problem [56], [62], [64]. In contrast, *Chameleon* NDA architecture shuns such inefficiency because they inherently offer more fine-grain DRAM memory requests than the traditional DIMM.

| Name | Description | # of Kernels | Iterative? | Replaced Execution Time | Shuffling Overhead |
|---|---|---|---|---|---|
| BP | Back propagation [13] | 4 | Yes | 99.9% | 1.6% |
| DISP | Disparity map [55] | 10 | No | 99.9% | 0.1% |
| HACC | Hardware accelerated cosmology code [1] | 2 | Yes | 99.9% | 1.5% |
| HS | Hotspot [13] | 2 | Yes | 99.9% | 1.1% |
| KM | K-means clustering [13] | 1 | Yes | 99.9% | 0.7% |
| LBM | Lattice-Boltzmann method fluid dynamics [54] | 1 | Yes | 99.9% | 1.3% |
| MRIG | Magnetic resonance imaging gridding [54] | 1 | No | 98.5% | 0.14% |
| OCN | Ocean movements [59] | 16 | Yes | 81.7% | 1.6% |
| SIFT | Scale-invariant feature transform [55] | 4 | No | 92.7% | 0.1% |
| SRAD | Speckle reducing anisotropic diffusion [13] | 3 | Yes | 99.9% | 0.2% |
| TRCK | Feature tracking [55] | 8 | No | 79.3% | 2.0% |

**TABLE II: Benchmarks used in our evaluation are from the San Diego Vision [55], Parboil [54], CORAL [1], SPLASH-2 [59], and Rodinia [13] benchmark suites.**

## V. Methodology

In this section, we describe our methodology such as evaluated benchmarks, processor, and accelerator specifications.

**Benchmark:** *Chameleon* benefits from the memory-intensive benchmarks with limited temporal locality that have simple operations repetitively applied in parallel to different data. Table II summarizes 11 big-data benchmarks from the San Diego Vision [55], Parboil [54], CORAL [1], SPLASH-2 [59], and Rodinia [13] benchmark suites that we use for evaluations. These highly parallel benchmarks can distribute their input sets across accelerators to exploit concurrency and localize most of memory accesses. This in turn reduces inter-accelerator communication, providing higher energy efficiency and speedup. Each benchmark has a different number of kernels. Except OCN and TRCK, more than 90% of total execution time is offloaded to accelerators.

Applications are decomposed into phases that are either executed in CPUs or accelerators. Table II shows the CPU execution time replaced by accelerators and the shuffling overhead attributed to the phase change process.

**Host Processor:** We take a four-way OoO processor as our baseline that has a 16-stage execution pipeline, which is configured similar to an Intel's Haswell processor [24] and simulate it using gem5. Table III tabulates the primary architecture parameters of the processor and its caches.

**Accelerator:** Although any programmable compute unit such as SIMD engines and GPU cores can be an accelerator for *Chameleon*, we chose CGRA due to its low control overhead (high energy efficiency), and integrate its model in gem5 in this paper; CGRA is a dataflow architecture where input data flow through configurable interconnects that connect functional units (FUs) in a desired manner (dataflow of given code), offering a much lower control overhead than SIMD engines and GPU cores. As we took a primitive in-house CGRA, we manually converted and mapped compute/data-intensive CPU code regions to CGRA code for each benchmark. However, the use of commercial CGRAs will allow us to use a good compiler (e.g., [41]) and offers better performance. Subsequently, we summarize the architecture, technology parameters, power consumption and area of the CGRA based on the specifications and analyses described in [20]. Table IV lists the details of a CGRA-style accelerator that is equipped with 64 functional units (FUs) that can support either integer or floating-point computations (similar to GPU execution units).

| Parameters | Values |
|---|---|
| Pipeline width | 4 |
| ROB/IQ/LQ/SQ entries | 128/36/48/32 |
| Integer and FP ALUs | Haswell-like |
| Int & FP physical Registers | 128 & 128 |
| Branch predictor | Tournament |
| BTB entries | 2,048 |
| I/D L1 caches | 32KB, 4-way for both |
| I/D L1 ports | 1/2 |
| L2 cache | 512KB, 8-way |
| Cache line size | 64 bytes |
| L1/L2 latency | 3/16 cycles |
| L1/L2 MSHRs | 10/16 |

**TABLE III: Key processor architecture parameters**

| Unit | Latency (cycles) | Count | Area ($\mu$m$^2$) | Energy per op. |
|---|---|---|---|---|
| INT ALU | 1 | 40 | 3,589 | 2.2 pJ |
| FP ALU | 5 | | 4,762 | 7.1 pJ |
| INT MUL | 2 | 20 | 6,507 | 13.1 pJ |
| FP MUL | 4 | | 6,565 | 11.3 pJ |
| INT DIV | 8 | 4 | 10,596 | 30.1 pJ |
| FP DIV | 9 | | 15,484 | 27.7 pJ |
| Switch | - | 81 | 4,236 | 1.11 pJ |

**TABLE IV: Design parameters of a CGRA-type accelerator with 64 32-bit FUs.**

In order to evaluate the area and power consumption of FUs and switches in our accelerators, the developed Verilog models of the FUs and the switches using Synopsys DesignWare building block IPs were used. Besides, TSMC 40 *nm* low-power standard-cell library on the Synopsys design compiler 2013 was used for the synthesis of the models. As shown in Table IV, the area of a 64-FU CGRA is $\sim$0.832 mm$^2$ when the models were optimized to be clocked at 800 MHz. The simple datapath and low operating frequency enabled CGRAs

| Parameters | Values |
|---|---|
| Number of banks | 16 |
| Row buffer size | 1KB |
| Clock frequency (1/tCK) | 1,200MHz |
| Cycle time (tRC) | 45.32ns |
| Activate to read/write time (tRCD) | 13.32ns |
| Access latency (tCL) | 13.32ns |
| Precharge latency (tRP) | 13.32ns |
| Activate to Activate between different bank groups (tCCD_S) | 4 tCK |
| Activate to Activate to the same bank group (tCCD_L) | 6 tCK |
| Four activate window (tFAW) | 21ns |
| Activate energy | 2.1nJ |
| Read/write energy without I/O | 14pJ/b |
| Off-chip I/O energy | 24pJ/b |

**TABLE V: Timing and energy parameters of a 8Gb DDR4-2400 ×8 DRAM device.**

to consume little dynamic power, which is ranged between 4 and 21 mW/mm$^2$. After consulting with a data buffer chip design company, we confirm that our CGRA of 0.832mm$^2$ can be sufficiently integrated on the buffer die without impacting the package size. Lastly, a memory controller within a CGRA is estimated to be ∼0.21 mm$^2$ and the peak dynamic power consumption is estimated to be 0.389 W based on McPAT [35].

**Accelerator in CPU (ACCinCPU):** For ACCinCPU, we take the same accelerator (model) as *Chameleon* and integrate it within the host processor in gem5 such that the accelerator shares the on-chip caches with the host processor and the coherency is supported [45], [57]. This baseline architecture can sometimes offer higher performance than NDA architectures when given applications exhibit high temporal locality in their accessed data [20]. This is because such applications greatly benefit from the on-chip caches that can provide requested data much faster and more efficiently than the DRAM devices. For fair comparison, we populate the same number of accelerators for both ACCinCPU and *Chameleon*.

**Memory:** DDR4-2400 DRAM devices with data I/O sizes of 8 bits (×8), which are common for DDR4, were used. In the ×8 interface, there are eight memory devices in a given rank (except the one for ECC) to form a 64-bit bus between memory devices and a memory controller. In general, DDR4 devices have a burst-length of 8, transferring 64 bytes of data in each memory transfer. Table V contains detailed timing parameters of DRAM subsystem used in our evaluation.

## VI. Experiment

### A. Performance Comparison

*Chameleon* does not rely on 3D/2.5D stacking technology unlike the past NDA architectures and instead places accelerators in the LRDIMM's DB devices that are packaged separately from DRAM devices. Nonetheless, *Chameleon* can still offer as competitive performance and energy efficiency as an NDA architecture that exploits 3D die-stacking. Figure 6 shows the relative performance of *Chameleon-d*, *-s*, and *-t* over the configuration with accelerators stacked atop DRAM devices (ACCinDRAM, which is NDA-1 in [20]) for the evaluated benchmarks. It also depicts the speedup of ACCinDRAM over the baseline 4-way out-of-order processor. As for *Chameleon-s*, we varied the number of DQ pins dedicated to data transfers. For example, *Chameleon-s* ×6 means that six out of eight DQ pins were used for data transfers in NDA mode.

We made the following key observations. First, *Chameleon-s* and *-t* outperformed *Chameleon-d* even if the former two share the DQ pins for C/A/DQ signals whereas the latter microarchitecture dedicates those DQ pins only for data transfers. This is because the decoupled architecture of *Chameleon-d* routes C/A signals from all accelerators within a LRDIMM through a single shared RCD device. This RCD device forwards the C/A signals through a bus so that the accelerators must compete with each other to occupy this limited bus bandwidth. Therefore, *Chameleon-d* achieved only 29% performance of ACCinDRAM on average of the evaluated benchmarks. However, it still performs 3.5× better than the baseline four-way out-of-order processor configuration.

Second, both *Chameleon-s* and *-t* performed close to ACCinDRAM. When we kept the data transfer rate of DQ pins the same as that of ACCinDRAM (called normal mode), the performance gap between *Chameleon-s* ×6 and ACCinDRAM was as small as 4% on MRIG, but on average of 11 evaluated benchmarks, it was 21%. The gear-up mode narrowed this performance gap noticeably. For example, on benchmarks such as HACC, KM, MRIG, SIFT, and TRCK, the performance difference between *Chameleon-s* ×6 and ACCinDRAM was less than 1%. On average, the performance of *Chameleon-s* is 3% better than that of *Chameleon-t* and stays within 4% of the performance of ACCinDRAM when the gear-up mode was utilized. In the subsequent experiments, the gear-up mode was employed.

Third, among the configurations of *Chameleon-s*, dedicating 6 DQ pins to data transfers (*Chameleon-s* ×6) gave the highest performance. With more DQ pins dedicated to data transfers, fewer cycles are needed per data burst, but at the same time more cycles are needed for an accelerator to send a C/A pair to its coupled DRAM devices, increasing both serialization and queuing delays of the C/A channel in NDA mode. With fewer DQ pins dedicated to data transfers, these trade-offs are reversed. Benchmarks with less locality in DRAM accesses, such as MRIG, demand more bandwidth for C/A delivery so that fewer DQ pins to data transfers are favored. On average of the evaluated benchmarks, *Chameleon-s* ×6 was 9% and 17% better than *Chameleon-s* ×5 and ×4, respectively, in performance.

Compared to ACCinDRAM, *Chameleon* microarchitectures consumed more energy because inter-package communication is needed for DRAM accesses in NDA mode, as depicted in Figure 7. *Chameleon-d* consumed noticeably more energy than others due to its lower performance. *Chameleon-t* achieved the highest energy efficiency on TRCK, but on the other benchmarks, *Chameleon-s* ×6 was the most energy-efficient

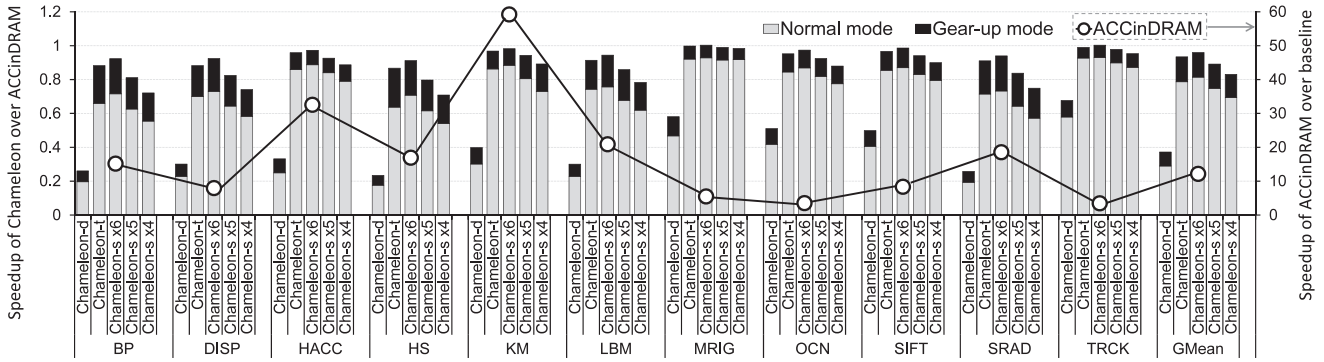



**Fig. 6: Performance of *Chameleon-d*, *-s*, and *-t* relative to the configuration with accelerators stacked atop DRAM devices (ACCinDRAM, NDA-1 in [20]) in the primary y-axis and performance of ACCinDRAM relative to the baseline host processor in the secondary y-axis; the dark stacked bars denote additional performance improvement after enabling the gear-up mode.**

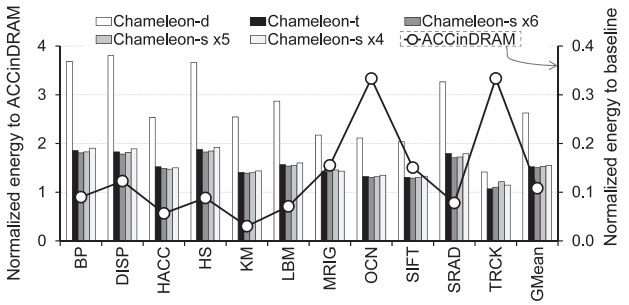



**Fig. 7: Energy consumption of *Chameleon-d*, *-s*, and *-t* relative to ACCinDRAM in the primary y-axis and energy consumption of ACCinDRAM over the baseline that does not utilize accelerators in the secondary axis.**

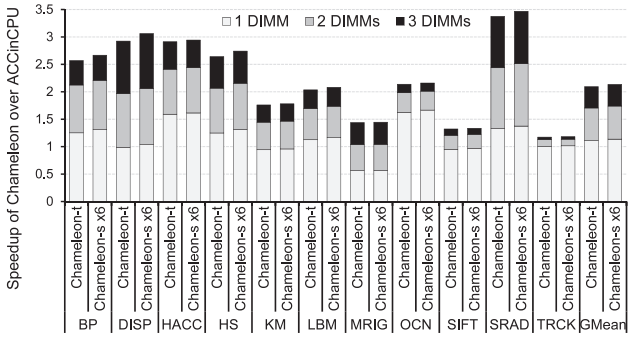



**Fig. 8: The relative performance of *Chameleon-s* ×6 and *Chameleon-t* over ACCinCPU as we increase the number of LRDIMMs per channel from one to two and three.**

configuration. Similar to the performance evaluation, all the *Chameleon* microarchitectures were still more energy efficient than the baseline.

### *B. Bandwidth Scaling*

At a given number of main memory channels, the performance of *Chameleon* microarchitectures scales with the number of LRDIMMs whereas that of ACCinCPU varies just marginally. Figure 8 shows the relative performance of *Chameleon-s* ×6 and *Chameleon-t* over ACCinCPU on the evaluated benchmarks when the number of LRDIMMs per channel is varied. In NDA mode, each LRDIMM of the *Chameleon* microarchitectures operates independently as illustrated in Figure 5. This makes the aggregate bandwidth provisioned to the accelerators proportional to the number of LRDIMMs. In contrast, accelerators located at a host processor (ACCinCPU) can exploit more banks from more populated LRDIMMs, but their performance is limited by the fixed channel bandwidth shared by all the LRDIMMs. Also bubbles due to the changes in the ownership of a memory channel negatively influence the performance of the accelerators. Benchmarks such as KM and MRIG better exploit CPU-side caches and hence ACCinCPU perform better than *Chameleon* microarchitectures. However, on most of the benchmarks *Chameleon* microarchitectures perform better and moreover, their performance scale with the number of LRDIMMs. On average, *Chameleon-s* ×6 and *Chameleon-t* perform 14% and 11% better than ACCinCPU, respectively, when one LRDIMM is populated. The performance gap between *Chameleon* and ACCinCPU, however, increases to 113% and 110% with 3 DIMMs populated per channel for *Chameleon-s* ×6 and *Chameleon-t*, respectively.

### *C. Bandwidth Utilization*

*Chameleon* utilizes a given main memory bandwidth more efficiently than ACCinCPU because the accelerators in DB devices operate independently so that the number of DRAM banks that can have different states is increased by the number of DB devices in an LRDIMM for *Chameleon*. Figure 9 compares the number of bytes accessed per row activation and page hit rate of *Chameleon-s* ×6 over that of ACCinCPU in the primary y-axis and DRAM energy of *Chameleon-s* ×6 over ACCinCPU in the secondary y-axis when one LRDIMM is populated per memory controller. Localized data memory of *Chameleon* architecture increases the average number of bytes

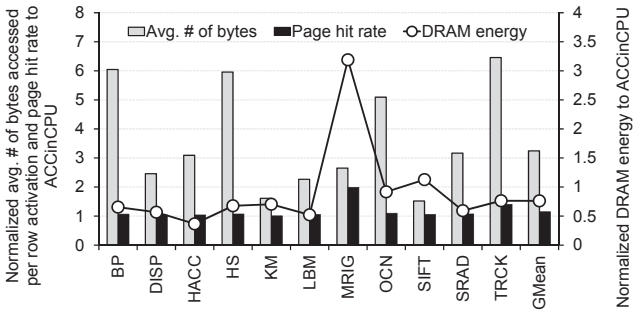



**Fig. 9: The relative number of bytes accessed per row activation and page hit rate of *Chameleon-s* ×6 over ACCinCPU (primary y-axis) and DRAM energy of *Chameleon-s* ×6 over ACCinCPU (secondary y-axis).**

accessed per row activation, therefore, fewer rows are activated required for *Chameleon* architecture to read and write data. A decrease in the number of row activations brings higher bandwidth utilization and reduces overall delay and energy associated with bank precharge and activations. *Chameleon-s* with 1 LRDIMM and gear-up mode accessed 6.1×, 3.1×, 6.0×, and 5.1× more bytes per row activation than ACCinCPU for BP, HACC, HS, and OCN, achieving 39%, 61%, 32%, and 67% higher performance, respectively. Lastly, the DRAM energy consumption of *Chameleon* architecture is lower than that of ACCinCPU architecture because of more efficient data access. For BP, HACC, HS, and OCN, *Chameleon-s* consumes 28%, 61%, 32%, and 30% lower DRAM energy than ACCinCPU, respectively.

## VII. Related Work

We categorize near-data processing studies that are closely related to our *Chameleon* by the topology between processing elements and their corresponding memory.

**On-die integration of processing elements and memory:** A large body of traditional processing-in-memory (PIM) studies integrates processing elements and DRAM onto a single die [15], [17], [19], [28], [39], [44], [47]. Those architectures pursue minimizing energy and maximizing throughput in data transfers between where it is processed and where it is contained. Computational RAM [19] unifies memory and processing elements by tying SIMD elements with DRAM sense amplifiers through a wide bus. Oskin et al. [44] propose active pages, which associate functional units that are implemented on reconfigurable fabric to numerous small DRAM pages. FlexRAM [28] works as plain DRAM devices but when needed, it utilizes a large number of single processing elements that are tightly interleaved with DRAM blocks. Intelligent RAM [47] integrates a vector processor with DRAM blocks on a single die to accelerate data-parallel operations. Mai et al. [39] propose a modular reconfigurable architecture dubbed Smart Memories, which connects many small processing elements, caches, and DRAM blocks with an on-chip network. Similarly, DIVA [17] incorporates multiple PIM tiles, each consisting of wide datapath and DRAM cells, to a host processor through an address translation unit. Although promising in motivations and concepts, integrating memory and logic in a die has a serious drawback in practicality; Logic tolerates leaky transistors pursuing high performance, whereas DRAM is primarily designed for high capacity and cell retention time with stringent cost constraints. Therefore, these PIM systems that integrate two different fabrication technologies incur high manufacturing costs and low yields [14], [46] not to mention additional design and verification complexities.

More recently, there are proposals to integrate logic with emerging non-volatile memory, such as STT-MRAM and PCM, to accelerate search operations [23] and enable more generic in-memory associate computing [22]. Automata Processor exploits massive bit-wise parallelism available in modern DRAM structures to parallelize automata processing by replacing conventional bitline sense amplifiers with state transition elements and substituting bus-based inter-bank datalines with more generic automata routing matrix [16]. These proposals are different from *Chameleon* in that they focus on accelerating a limited set of algorithms using specialized languages.

**Connecting memory and accelerators with TSVs:** 3D stacking enables DRAM layers to be stacked atop a logic layer through high bandwidth and highly energy-efficient TSVs, opening up new opportunities for local/near data processing [37], [48], [49], [63], [65]. For example, HMC [48] has MCs and ALUs for simple atomic operations within its logic layer. Smart memory cubes [7] extends these simple ALUs in HMC into more generic PIM. There are other proposals to stack these DRAM layers over application-specific integrated circuits [65] and GPGPU extenders [63]. NDA [20] is closest to our *Chameleon* architecture, which focuses on utilizing conservative 3D-stacking without demanding a complete redesign of commodity DRAM devices and their processor-memory interface. In contrast to these products and proposals that integrate processing elements within DRAM dies, place accelerators within MCs, or 3D-stack them in a single package, *Chameleon* is unique in that it locates accelerators in DB devices, which are separately packaged in a LRDIMM for better signal integrity, obviating need for still costly 3D integration and exploiting better energy efficiency and throughput scaling due to the fact that accelerators and DRAM devices are located at the same memory modules.

**Connecting memory and accelerators on a module:** An NDC-Module integrates lightweight processors and LPDDR2 ×32 devices onto an add-in daughter card modules and multiple NDC-Modules can be connected on the motherboard [49]. NDC-Module architecture inspired *Chameleon* to scale the memory bandwidth exposed to accelerators with more *Chameleon*-enabled DIMMs, but *Chameleon* is architected to conform with existing DDR4 and LRDIMM interfaces to support large capacity, whereas NDC-Module is not.

## VIII. Conclusion

We have proposed *Chameleon* – a pragmatic near-DRAM acceleration (NDA) architecture leveraging the current DDR4

LRDIMM architecture. Unlike prior NDA architectures, *Chameleon* places accelerators in DB devices of LRDIMMs, shunning the use of costly 3D-stacking technology – one of critical concerns of processing-in memory. We proposed *Chameleon-t* and *Chameleon-s* – two microarchitectures to abate a unique constraint imposed by the DDR4 LRDIMM architecture, wherein the standard data buffer devices do not provide dedicated pins for C/A signals. To compensate the bandwidth lost by sharing data pins for sending C/A signals, we proposed the gear-up mode that achieves a higher data transfer rate between the DB and DRAM devices, enabled by superior signaling conditions in NDA mode. Between the two microarchitectures, *Chameleon-s* performed the best on average of the evaluated data-intensive benchmarks because it did not suffer from penalties due to frequent changes of DQs' owners. Despite the penalties in latency, bandwidth, and energy consumption due to communicating across packages, we demonstrated through evaluation that *Chameleon* still offers as competitive performance and energy efficiency as an NDA architecture relying on 3D-stacking technology. Our experiment shows that our *Chameleon-s* system can deliver 96% and 70% geo-mean performance and energy benefits of the prior NDA architecture. Lastly, we discover two aspects which help the NDA architectures built upon traditional (LR)DIMMs outperform the traditional architecture integrating accelerators within the host processor itself (ACCinCPU). Those are, higher aggregate bandwidth for accelerators and more efficient bandwidth utilization, which are not explicitly explored and analyzed by the prior study. Overall, our experiments exhibit that the *Chameleon-s* system with one, two, and three LRDIMMs can offer 14%, 74%, and 113% higher geo-mean performance than ACCinCPU.

## ACKNOWLEDGMENT

The authors would like to thank Hyunyoon Cho for SPICE modeling. This work was supported in part by generous grants from Samsung Semiconductor (2016-03835), NSF (CNS-1217102), DARPA (HR0011-12-2-0019), and MOTIE/KSRC (Future Semiconductor Device Technology Development Program-10044735). Nam Sung Kim has a financial interest in AMD and Samsung Electronics.

## REFERENCES

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

[2] "Samsung DDR4 SDRAM." [Online]. Available: http://www.samsung.com/global/business/semiconductor/product/dram

[3] "JEDEC Standard: DDR2 SDRAM Fully Buffered DIMM (FBDIMM) Design Specification," 2007.

[4] "JEDEC Standard: DDR4 SDRAM," 2012.

[5] "JEDEC Standard: DDR4 SDRAM Load Reduced DIMM (LRDIMM) Design Specification," 2014.

[6] "JEDEC Standard: Low Power Double Data Rate 4 (LPDDR4)," 2014.

[7] E. Azarkhish, D. Rossi, I. Loi, and L. Benini, "A Logic-base Interconnect for Supporting Near Memory Computation in the Hybrid Memory Cube," in *Workshop on Near-Data Processing*, Dec 2014.

[8] S.-J. Bae, Y.-S. Sohn, K. ll Park, K.-H. Kim, D.-H. Chung, J.-G. Kim, S.-H. Kim, M.-S. Park, J.-H. Lee, S.-Y. Bang, H.-K. Lee, I.-S. Park, J.-S. Kim, D.-H. Kim, H.-R. Kim, Y.-J. Shin, C.-G. Park, G.-S. Moon, K.-W. Yeom, K.-Y. Kim, J.-Y. Lee, H.-J. Yang, S.-J. Jang, J. S. Choi, Y.-H. Jun, and K. Kim, "A 60nm 6Gb/s/pin GDDR5 Graphics DRAM with Multifaceted Clocking and ISI/SSN-Reduction Techniques," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2008.

[9] A. Basu, J. Gandhi, J. Chang, M. D. Hill, and M. M. Swift, "Efficient Virtual Memory for Big Memory Servers," in *ISCA*, June 2013.

[10] S. Borkar, "Role of Interconnects in the Future of Computing," *IEEE Journal of Lightwave Technology*, vol. 31, no. 24, Dec 2013.

[11] D. W. Chang, G.-s. Byun, H. Kim, M. Ahn, S. Ryu, N. Kim, and M. Schulte, "Reevaluating the Latency Claims of 3D Stacked Memories," in *IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2013.

[12] D. W. Chang, Y. H. Son, J. Ahn, H. Kim, M. Ahn, M. J. Schulte, and N. S. Kim, "Dynamic bandwidth scaling for embedded DSPs with 3D-stacked DRAM and wide I/Os," in *ICCAD*, 2013.

[13] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *IISWC*, Oct 2009.

[14] M. L. Chu, N. Jayasena, D. P. Jang, and M. Ignatowski, "High-level Programming Model Abstractions for Processing in Memory," in *Workshop on Near-Data Processing*, Dec 2013.

[15] M. F. Deering, S. A. Schlapp, and M. G. Lavelle, "FBRAM: a New Form of Memory Optimized for 3D Graphics," in *ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, Jul 1994.

[16] P. Dlugosch, D. Brown, P. Glendenning, M. Leventhal, and H. Noyes, "An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 25, no. 12, Dec 2014.

[17] J. Draper, J. Chame, M. Hall, C. Steele, T. Barrett, J. LaCoss, J. Granacki, J. Shin, C. Chen, C. W. Kang, I. Kim, and G. Daglikoca, "The Architecture of the DIVA Processing-in-memory Chip," in *ICS*, Jun 2002.

[18] J. Ekanayake, S. Pallickara, and G. Fox, "Mapreduce for Data Intensive Scientific Analyses," in *IEEE International Conference on eScience*, Dec 2008.

[19] D. G. Elliott, W. M. Snelgrove, and M. Stumm, "Computational RAM: A Memory-SIMD Hybrid and its Application to DSP," in *IEEE Custom Integrated Circuits Conference (CICC)*, May 1992.

[20] A. Farmahini-Farahani, J. Ahn, K. Morrow, and N. S. Kim, "NDA: Near-DRAM Acceleration Architecture Leveraging Commodity DRAM Devices and Standard Memory Modules," in *HPCA*, Feb 2015.

[21] B. Ganesh, A. Jaleel, D. Wang, and B. Jacob, "Fully-Buffered DIMM Memory Architectures: Understanding Mechanisms, Overheads and Scaling," in *HPCA*, Feb 2007.

[22] Q. Guo, X. Guo, R. Patel, E. Ipek, and E. Friedman, "AC-DIMM: Associative Computing with STT-MRAM," in *ISCA*, Jun 2013.

[23] Q. Guo, X. Guo, Y. Bai, and E. Ipek, "A Resistive TCAM Accelerator for Data-intensive Computing," in *MICRO*, Dec 2011.

[24] P. Hammarlund, A. Martinez, A. Bajwa, D. Hill, E. Hallnor, H. Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar, R. Osborne, R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupaty, S. Jourdan, S. Gunther, T. Piazza, and T. Burton, "Haswell: The Fourth-Generation Intel Core Processor," *IEEE Micro*, vol. 34, no. 2, Mar 2014.

[25] R. Ho, K. Mai, and M. Horowitz, "The Future of Wires," *Proceedings of the IEEE*, vol. 89, no. 4, Apr 2001.

[26] Inphi, "Introducing LRDIMM - A New Class of Memory Modules." [Online]. Available: https://www.inphi.com/products/whitepapers/Inphi_LRDIMM_whitepaper_Final.pdf

[27] Y.-C. Jang, H. Chung, Y. Choi, H. Park, J. Kim, S. Lim, J. Sunwoo, M.-S. Park, H.-S. Kim, S.-Y. Kim, Y.-S. Lee, W.-S. Kim, J.-B. Lee, J. Yoo, and C. Kim, "BER Measurement of a 5.8-Gb/s/pin Unidirectional Differential I/O for DRAM Application with DIMM Channel," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 11, Nov 2009.

[28] Y. Kang, W. Huang, S.-M. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik, and J. Torrellas, "FlexRAM: Toward an Advanced Intelligent Memory System," in *IEEE International Conference on Computer Design (ICCD)*, Oct 1999.

[29] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "GPUs and the Future of Parallel Computing," *IEEE Micro*, vol. 31, no. 5, Sep 2011.

[30] G. Kim, J. Kim, J. H. Ahn, and J. Kim, "Memory-centric System Interconnect Design with Hybrid Memory Cubes," in *International Conference on Parallel Architectures and Compilation Techniques(PACT)*, 2013.

[31] Y. Kim, M. Papamichael, O. Mutulu, and M. Harchol-Balter, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, Dec 2010.

[32] R. Landaverde, T. Zhang, A. Coskun, and M. Herbordt, "An Investigation of Unified Memory Access Performance in CUDA," in *IEEE High Performance Extreme Computing Conference (HPEC)*, Sep 2014.

[33] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, and S. Hong, "25.2 A 1.2V 8Gb 8-channel 128GB/s High-bandwidth Memory (HBM) Stacked DRAM with Effective Microbump I/O Test Methods Using 29nm Process and TSV," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2014.

[34] D. Lee, Y. Kim, G. Pekhimenko, S. Khan, V. Seshadri, K. Chang, and O. Mutlu, "Adaptive-latency DRAM: Optimizing DRAM Timing for the Common-case," in *HPCA*, 2015.

[35] S. Li, J. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 10, no. 1, Apr 2013.

[36] G. Loh, N. Jayasena, M. Oskin, M. Nutter, D. Roberts, M. Meswani, D. Zhang, and M. Ignatowski, "A Processing in Memory Taxonomy and a Case for Studying Fixed-function PIM," in *Workshop on Near-Data Processing*, Dec 2013.

[37] G. H. Loh, "3D-Stacked Memory Architectures for Multi-core Processors," in *ISCA*, Jun 2008.

[38] D. Luebke, "CUDA: Scalable Parallel Programming for High-performance Scientific Computing," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, May 2008.

[39] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz, "Smart Memories: A Modular Reconfigurable Architecture," in *ISCA*, Jun 2000.

[40] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz, "Towards Energy-proportional Datacenter Memory with Mobile DRAM," in *ISCA*, 2012.

[41] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: an architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in *Field Programmable Logic and Application, 13th International Conference*, 2003.

[42] Micron, "Micron Z80A 4Gb DDR4 x8 Package DQ S-parameters," 2015. [Online]. Available: http://www.micron.com/media/documents/products/sim-model/dram/z80a_x8_dq_sparam.zip

[43] R. Oh, B. Lee, S.-W. Shin, W. Bae, H. Choi, I. Song, Y.-S. Lee, J.-H. Choi, C.-W. Kim, S.-J. Jang, and J. S. Choi, "Design Technologies for a 1.2V 2.4Gb/s/pin High Capacity DDR4 SDRAM with TSVs," in *IEEE Symposium on VLSI Circuits*, Jun 2014.

[44] M. Oskin, F. Chong, and T. Sherwood, "Active Pages: A Computation Model for Intelligent Memory," in *ISCA*, Jun 1998.

[45] A. Parashar, M. Pellauer, M. Adler, B. Ahsan, N. Crago, D. Lustig, V. Pavlov, A. Zhai, M. Gambhir, A. Jaleel, R. Allmon, R. Rayess, S. Maresh, and J. Emer, "Triggered Instructions: A Control Paradigm for Spatially-programmed Architectures," in *ISCA*, Jun 2013.

[46] D. Patterson, K. Asanovic, A. Brown, R. Fromm, J. Golbus, B. Gribstad, K. Keeton, C. Kozyrakis, D. Martin, S. Perissakis, R. Thomas, N. Treuhaft, and K. Yelick, "Intelligent RAM (IRAM): The Industrial Setting, Applications, and Architectures," in *IEEE International Conference on Computer Design (ICCD)*, Oct 1997.

[47] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "A Case for Intelligent RAM," *IEEE Micro*, vol. 17, no. 2, Mar 1997.

[48] J. T. Pawlowski, "Hybrid Memory Cube," in *Hot Chips*, Aug 2011.

[49] S. H. Pugsley, J. Jestes, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis, and F. Li, "Comparing Implementations of Near-Data Computing with In-Memory MapReduce Workloads," *IEEE Micro*, vol. 34, no. 4, Jul 2014.

[50] S. H. Pugsley, J. Jestes, H. Zhang, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis, and F. Li, "Ndc: Analyzing the impact of 3d-stacked memory+ logic devices on mapreduce workloads," in *ISPASS*, 2014.

[51] R. Sampson, M. Yang, S. Wei, C. Chakrabarti, and T. F. Wenisch, "Sonic Millip3De: A Massively Parallel 3D-stacked Accelerator for 3D Ultrasound," in *HPCA*, Feb 2013.

[52] A. Sodani, "Rethinking Design Metrics for Datacenter DRAM," in *ACM International Symposium on Memory Systems (MEMSYS)*, 2015, pp. 162–163.

[53] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems," *IEEE Computing in Science & Engineering*, vol. 12, no. 3, May 2010.

[54] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-M. Hwu, "Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing," *Center for Reliable and High-Performance Computing*, Mar 2012.

[55] S. K. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. B. Taylor, "SD-VBS: The San Diego Vision Benchmark Suite," in *IISWC*, Oct 2009.

[56] F. A. Ware and C. Hampel, "Improving Power and Data Efficiency with Threaded Memory Modules," in *IEEE International Conference on Computer Design (ICCD)*, Oct 2006.

[57] M. Watkins and D. Albonesi, "ReMAP: A Reconfigurable Heterogeneous Multicore Architecture," in *MICRO*, Dec 2010.

[58] K. Wilcox, D. Akeson, H. Fair, J. Farrell, D. Johnson, G. Krishnan, H. Mclntyre, E. McLellan, S. Naffziger, R. Schreiber, S. Sundaram, and J. White, "A 28nm x86 APU Optimized for Power and Area Efficiency," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2015.

[59] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *ISCA*, Jun 1995.

[60] D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan, "BOOM: Enabling Mobile Memory Based Low-Power Server DIMMs," in *ISCA*, Jun 2012.

[61] D. H. Yoon and M. Erez, "Virtualized and Flexible ECC for Main Memory," in *ASPLOS*, Mar 2010.

[62] D. H. Yoon, M. K. Jeong, and M. Erez, "Adaptive Granularity Memory Systems: A Tradeoff between Storage Efficiency and Throughput," in *ISCA*, Jun 2011.

[63] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski, "TOP-PIM: Throughput-oriented Programmable Processing in Memory," in *ACM International Symposium on High-performance Parallel and Distributed Computing (HPDC)*, Jun 2014.

[64] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-Rank: Adaptive DRAM Architecture for Improving Memory Power Efficiency," in *MICRO*, Nov 2008.

[65] Q. Zhu, B. Akin, H. E. Sumbul, F. Sadi, J. C. Hoe, L. Pileggi, and F. Franchetti, "A 3D-stacked Logic-in-memory Accelerator for Application-Specific Data Intensive Computing," in *IEEE International 3D Systems Integration Conference (3DIC)*, Oct 2013.