2017 IEEE International Symposium on High Performance Computer Architecture

# SOUP-N-SALAD: Allocation-oblivious Access Latency Reduction with Asymmetric DRAM Microarchitectures

Yuhwan Ro*, Hyunyoon Cho*†, Eojin Lee*, Daejin Jung*, Young Hoon Son†, Jung Ho Ahn* and Jae W. Lee*
*Seoul National University, †Samsung Electronics
{gajh, jaewlee}@snu.ac.kr

***Abstract*—Memory access latency has a significant impact on application performance. Unfortunately, the random access latency of DRAM has been scaling relatively slowly, and often directly affects the critical path of execution, especially for applications with insufficient locality or memory-level parallelism. The existing low-latency DRAM organizations either incur significant area overhead or burden the software stack with non-uniform access latency. This paper proposes two microarchitectural techniques to provide *uniformly low* access time over the entire DRAM chip. The first technique is SALAD, a new DRAM device architecture that provides *s*ymmetric *a*ccess *l*atency with *a*symmetric *D*RAM bank organizations. Because local regions have lower data transfer time due to their proximity to the I/O pads, SALAD applies high aspect-ratio (i.e., low-latency) mats only to remote regions to offset the difference in data transfer time, resulting in symmetrically low latency *across* regions. The second technique is SOUP (*s*kewed *o*rganization of $\mu$banks with *p*ipelined accesses), which leverages asymmetry in column access latency *within* a region due to non-uniform distance to the column decoders. By starting I/O transfers as soon as data from near cells arrive, instead of waiting for the entire column data, SOUP further saves two memory clock cycles for column accesses for *all* regions. The resulting design, called *SOUP-N-SALAD*, improves IPC and EDP by 9.6% (11.2%) and 18.2% (21.8%) over the baseline DDR4 device, respectively, for memory-intensive SPEC CPU2006 workloads without any software modifications, while incurring only 3% (6%) area overhead.**

***Keywords*-DRAM; symmetric access latency; asymmetric DRAM bank organizations; skewed and pipelined accesses;**

## I. Introduction

Memory access latency has a significant impact on application performance. The latency of DRAM, which forms the foundation of main memory, has been scaling much more slowly than its bandwidth and capacity, to be still over 100 CPU clock cycles [34], [41]. Latency-hiding techniques, such as caching, prefetching, and out-of-order execution, do not completely resolve this problem, especially for applications with insufficient locality or memory-level parallelism.

Most existing low-latency DRAM organizations either incur significant area overhead [14] or burden the hardware design and the software stack with non-uniform access latency [5], [7], [25], [41]. For example, Reduced Latency DRAM (RLDRAM [14]) provides uniformly low access latency at a high area cost (40-80% larger than the baseline DDR2 DRAM device [21]). To amortize the high device cost, Chatterjee et al. [5] propose to use RLDRAM only for critical words and low-cost (LP)DDR DRAM for the rest. Recently, novel DRAM organizations have been proposed to alleviate the area overhead by applying latency reduction techniques only to a portion of the device (CHARM [41]), by using an isolation transistor to split a long bitline into near and far segments (Tiered-Latency DRAM [25]), and by trading the capacity of a device with its access time by combining multiple DRAM cells into one logical cell (Multiple Clone Row DRAM [7]).

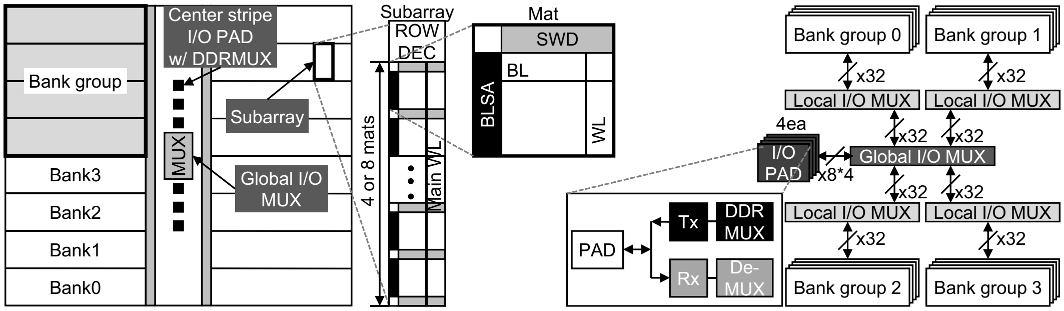
However, those new NUMA memory devices and systems often require intrusive modifications along the memory access path in hardware, software, or both. For example, CHARM, a state-of-the-art organization, reduces both activate (tRCD) and precharge (tRP) time only for a small region close to the I/O pads by making fewer DRAM cells share a bitline. This makes CHARM a two-level NUMA device comprised of the fast local region and the slow remote region, and hence requires modifications to the memory controller. Furthermore, software needs to be rewritten to identify hot memory pages and allocate them to the fast region. Without such *criticality-aware* memory allocation, it is difficult for CHARM to achieve robust performance.

To address this problem, this paper proposes two microarchitectural techniques to provide *uniformly low* access time over an entire DRAM chip, and hence robust performance without requiring sophisticated, criticality-aware memory allocation. The first technique is SALAD, a new DRAM device architecture that provides symmetric access latency with asymmetric DRAM bank organizations. SALAD leverages the asymmetric bank structure of CHARM in an opposite way—it applies high aspect-ratio (i.e., low-latency) mats to remote regions, instead of local regions, thus reducing the access latency (tAC) of a whole device. This balanced latency reduction is achieved by different regions reducing components of tAC, which is a sum of activation time (tRCD) and column access latency (tCL). Remote regions have lower bitline capacitance, and hence lower tRCD; local regions have lower I/O transfer time due to shorter topological distance, and hence lower tCL. By providing

Y. Ro and H. Cho are the co-first authors. J. Ahn and J. W. Lee are the corresponding authors. The work was done when the authors were at Seoul National University.

2378-203X/17 $31.00 © 2017 IEEE
DOI 10.1109/HPCA.2017.31

IEEE computer society



(a) DRAM floorplan

(b) Simplified data I/O flow block diagram

Figure 1: Contemporary DDR4 SDRAM device organization. (a) A DRAM device takes a hierarchy of bank groups, banks, subarrays, and mats, where a mat is a basic building block consisting of 2D array of DRAM cells. (b) Each bank group owns a local I/O MUX and all the banks in a device share a global I/O MUX. 32-bit data is distributed across 4 I/O pads (assuming x4 configuration) with 8 bits per each using DDRMUX (i.e., burst length is 8).

uniformly low access time, SALAD not only obviates the need for expensive software changes but also improves bank-level parallelism by better spreading memory accesses across all banks, instead of a handful of hot banks with fast mats.

The second technique is SOUP (skewed organization of $\mu$banks with pipelined accesses), leveraging latency asymmetry within a region to provide better-than-worst-case latency for column accesses. Each region is divided into four *segments* depending on the distance to the column decoders. For column accesses, data from the nearest segment arrive at the global I/O MUX the earliest, and the farthest one the latest. Unlike the conventional DDR4 device, which waits for the entire column data to arrive, SOUP starts I/O transfers as soon as the data from the nearest segment arrive, hence effectively pipelining cell array accesses and I/O operations. This saves two memory clock cycles (tCK).

Our evaluation with single- and multi-programmed workloads using SPEC CPU2006 benchmarks [12] shows that, even with data *criticality-oblivious* mapping, the resulting design, called SOUP-N-SALAD, improves IPC and EDP by 9.6% (11.2%) and 18.2% (21.8%) while incurring 3% (6%) area overhead. This compares favorably to other alternative techniques: 1.6% (3.8%) IPC and 3.2% (7.8%) EDP improvements with all regions employing high aspect-ratio mats, and 5.3% (6.3%) and 11.4% (6.3%) with CHARM augmented by hot page-aware memory allocation.

In summary, this paper makes the following contributions:

- We propose SALAD, a new asymmetric DRAM device organization with symmetrically low access latency (tAC) over the entire device, providing robust performance even with criticality-oblivious memory allocation.
- We propose SOUP, which leverages latency asymmetry at a finer granularity within a region to save two memory clock cycles for CAS latency.
- We quantify the system-wide benefits of SOUP-N-SALAD in terms of IPC and EDP using single- and multi-programmed, and multi-threaded workloads.

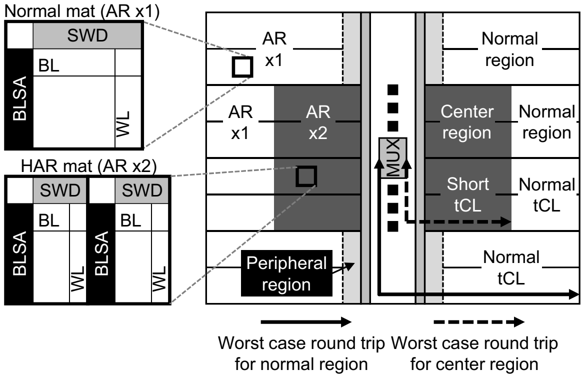
## II. Background and Motivation

### A. DDR4 DRAM Device Organization

Figure 1(a) depicts the floorplan of a contemporary DRAM device employing a hierarchical structure. A DRAM device consists of multiple banks (e.g., 8 for DDR3 and 16 for DDR4), each with thousands of *mats* arranged in two dimensions. A mat is a 2D array of cells with hundreds of rows and columns (e.g., 512×512 bits), where a row is connected to a sub-wordline and a column shares a local bitline (BL). A bitline sense amplifier (BLSA) detects the small voltage difference developed by charge sharing between a cell and a BL within a selected row. The data latched at the activated BLSAs are multiplexed into 32-bit global datalines, which are in turn connected to the I/O pads placed at the center of a device through inter-bank datalines and I/O MUXes (see Figure 1(b)). With DDRMUX, a part of the I/O pad, data is transferred through a 8-bit parallel bus. Since an I/O pad has only one transmission line per DQ (data pin) and transmitting per-pin bandwidth is much higher than bandwidth per internal wire (e.g., 2.4 Gbit/s/pin vs. 0.3 Gbit/s per inter-bank dataline wire for DDR4-2400), it is necessary to serialize data for boosting its speed. Accordingly, DDRMUX serializes 8 bit parallel data into burst length with 8, then transmits data through the channel. Unlike DDR3, DDR4 adopts the bank group architecture to maximize internal bandwidth.

An array of mats sharing a main wordline (WL) in a bank is called *subarray*. While all mats in a subarray operate in tandem, multiple subarrays in a bank can operate independently. Therefore, ideally, DRAM operations that do not utilize global datalines and inter-bank datalines, such as activation, precharge, and refresh, targeting different subarrays can be overlapped in a bank [24].

Similar to previous generations, a typical DRAM access cycle with DDR4 devices takes the following steps. First, a bank is idle with BLs precharged at $(V_{CC}+V_{SS})/2$, where



(a) Normal and HAR mat (b) CHARM architecture

Figure 2: CHARM device organization [41] places HAR (high-aspect-ratio) mats to the center banks of a device to substantially reduce both access time (tAC) and cycle time (tRC) of a small (one fourth here) region of a device.

$V_{CC}$ and $V_{SS}$ are the voltages representing one and zero, respectively. Once an activate (`ACT`) command is sent, it takes tRCD to activate the target row. Then a read (`RD`) or write (`WR`) command is applied to access the requested column from the activated row through I/O pins, which takes tCL. Note that the access time of a device (tAC) is defined as a sum of the minimum `ACT` to `RD`/`WR` time (tRCD) and the `RD` command to the first data at the I/O time (tCL). In the meanwhile, activated BLSAs continue to develop the voltage difference between a pair of differential BLs to be $V_{CC}$ and $V_{SS}$, respectively, which takes tRAS (activate to precharge delay). After tRAS has lapsed, the original voltage level of the cell is restored, and the bank is ready to accept a precharge command (`PRE`). Upon receiving `PRE`, the differential BLs are precharged back at ($V_{CC}$+$V_{SS}$)/2, which takes the precharge time (tRP). The cycle time of a device (tRC) is defined as a sum of tRAS and tRP.

### *B. Low-Latency DRAM Organizations*

There are existing DRAM organizations that provide uniformly low access latency by employing *high-aspect-ratio (HAR)* mats. A HAR mat reduces the BL capacitance by populating fewer WLs per mat, and hence connecting fewer cells per BL to yield lower activation (tRCD) and precharge (tRP) time as both tRCD and tRP are directly proportional to BL (dis)charging time. Reduced Latency DRAM (RLDRAM) [14] uses HAR mats on an entire device for uniformly low latency, but at a significant area cost due to duplicated BLSAs and increased wiring overheads. For example, the die size of a DDR2 RLDRAM is reported to be 40-80% larger than the baseline DDR2 DRAM [21]. To mitigate this overhead, Chatterjee et al. [5] proposed to use RLDRAM devices only for critical words and low-cost (LP)DDR devices for the rest. However, their approach requires extra memory controllers to support multiple types of DRAM devices and modifications to the MSHR to support fragmented transfers of a cache line over multiple channels.

Recently, novel DRAM organizations with better-than-worst-case delays for a portion of the chip have been proposed to alleviate the high device cost. CHARM [41] replaces mats at the center region of the chip with HAR mats (see Figure 2). The center region has a significantly lower latency than the rest of the chip not only for smaller tRCD and tRP with HAR mats but also for shorter data transfer time (tCL) due to their physical proximity to the I/O pads. Tiered-Latency DRAM (TLDRAM) [25] inserts an isolation transistor on each BL and splits cells in a mat into near and far segments. When activating or precharging the rows in a near segment, BL capacitance is reduced by cutting off far segment side BLs, hence yielding lower tRCD and tRP. Multiple Clone Row DRAM [7] speeds up accesses by multiplying cell capacitance. It combines multiple DRAM cells into one logical cell to increase (logical) cell capacitance at the cost of reduced memory capacity. All three examples above make DRAM a two-level NUMA device with asymmetric access latencies.

However, these emerging asymmetric DRAM devices require intrusive modifications along the memory access path in software, hardware, or both. Frequently accessed memory pages (hot pages) must be identified ahead of or during execution and allocated to fast regions. Those hot pages can be annotated during application development or profiled by test runs. However, memory access behaviors of many applications change dynamically based on input datasets, limiting the effectiveness of the annotation/profiling-based techniques. Besides, CHARM [41], a state-of-the-art asymmetric DRAM organization, concentrates memory accesses only to center banks with HAR mats, thus poorly exploiting bank-level parallelism. This is particularly problematic when an application has low spatial locality because modern DRAM relies on concurrently utilizing multiple banks to achieve reasonable random access performance.

## III. SALAD: Providing Uniformly Low DRAM Access Time

To address the aforementioned limitations of DRAM devices with non-uniform access latency (tAC), we propose SALAD, a novel DRAM device organization with the following design objectives:

- *Low average latency*: The average memory access latency should be low to provide robust performance even for applications with insufficient spatial locality or memory-level parallelism.
- *Uniform access latency*: The DRAM device should provide uniform access latency to achieve robust performance without intrusive modifications to the software stack.
- *Low hardware cost*: The cost of hardware should be minimal in terms of area and power.

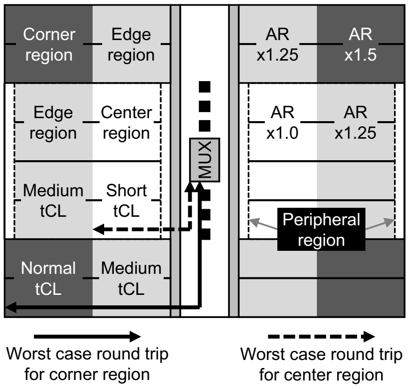


Figure 3: SALAD organization with HAR mats employed at edge and corner regions.

## A. SALAD Organization

Figure 3 illustrates the device architecture of SALAD (**S**ymmetric **A**ccess **L**atency with **A**symmetric **D**RAM). SALAD divides a DRAM device into three types of regions according to the distance from its I/O pads: 4 center, 8 edge, and 4 corner regions.[1] Similar to CHARM, SALAD leverages the ideas of 1) non-uniform I/O accesses to those regions [22] to reduce the read/write latency of center regions (i.e., tCL) and 2) increasing the aspect ratio (AR) of a portion of DRAM subarrays to reduce tRCD and tRP. However, unlike CHARM, SALAD applies HAR mats to edge and corner regions, instead of center regions that have the lowest transmission latency (tCL) due to their shorter physical distance to the multiplexers and I/O pads. Throughout this paper, we assume an 8Gb DDR4-2400 device [31] with 16 banks and 4 bank groups as a baseline. Although the three region types have different values for tCL, tRCD, and tRC, they have comparable access latencies (tAC). The center regions have mats with the default AR (512×512 cells per mat), and the eight edge regions have mats with a higher AR (lower tRCD) to compensate longer transmission delay (yielding higher tCL) of inter-bank command/address/data signals. The AR of the mats in the four corner regions is even higher. Because signals experience the longest transmission delay to the corner regions, tCL for those regions is the highest (i.e., same as baseline DRAM). Thus, we apply mats with the highest AR to the corner regions to compensate for an increase in tCL with reduced tRCD.

Figure 4 compares the access latency (tAC) of SALAD and CHARM. SALAD has different tAC values compared to CHARM which reduces tAC significantly only for a few center regions per device (Figure 4(e)); tAC for the rest of the chip remains unchanged in CHARM (Figure 4(a)). By contrast, all three regions in SALAD have (nearly) symmetric tAC values (Figure 4(b)–(d)) as a higher tCL at the edge and corner regions is compensated by a lower tRCD with HAR mats. Therefore, SALAD is much less sensitive to the quality of hot page allocation than CHARM, hence achieving more robust performance.

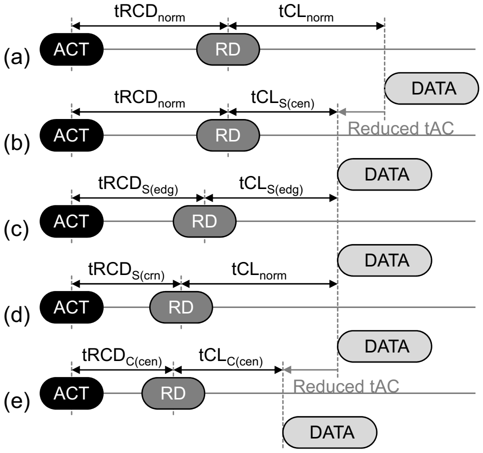


Figure 4: The timing diagram of access latency (tAC) for CHARM/SALAD (drawn not to scale). (a) Edge regions in CHARM (nominal tAC); (b) Center (cen) regions in SALAD; (c) Edge (edg) regions in SALAD; (d) Corner (crn) regions in SALAD; (e) Center (cen) regions in CHARM.

Table I tabulates the energy and timing parameters of the proposed SALAD architecture with area constraints of up to 3% and 6% increases (modeling methodology detailed in Section V). To satisfy these constraints, SALAD employs HAR mats with smaller ARs than CHARM as both edge and corner regions use HAR mats in SALAD (with about 75% area coverage in Figure 3), but only center regions in CHARM (with about 25% area coverage in Figure 2). More specifically, edge and corner regions in SALAD have ARs of ×1.25 and ×1.5, respectively, for a 3% area overhead, and ×1.5 and ×2.0 for a 6% overhead. By contrast, center regions in CHARM have ARs of ×2.0 and ×4.0 for 3% and 6% overheads, respectively. HAR mats significantly reduce activate/precharge times but slightly increase read and write energy due to longer signaling paths for command, address, and data. According to our SPICE modeling, the time saving benefit significantly outweighs the RD/WR energy cost (less than 0.7%), thus achieving net gains in energy efficiency.

Due to asymmetry in ARs among regions, the size of a bank depends on which type of region it belongs to. The spare area resulting from the mismatch in bank sizes is filled with peripheral resources (see Figure 3), such as I/O pads, decoupling capacitors, and charge-pump circuitry, similar to CHARM. This spare area can also be used for TSVs to stack multiple DRAM dies [35] to improve area efficiency.

## B. SALAD Operations

SALAD effectively improves the performance of applications even if hot pages do not exist or are not identified.

[1] Every bank interleaves two kinds of regions due to the bank placement of modern DRAM (see Figure 1(a)). Section IV elaborates it further.

| | Baseline | All-HAR | | CHARM | | | | SALAD | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | All reg. | | Center reg. | | The others | | Corner reg. | | Edge reg. | | Center reg. | |
| Area overhead | 0% | 3% | 6% | 3% | 6% | 3% | 6% | 3% | 6% | 3% | 6% | 3% | 6% |
| Aspect Ratio (AR) | ×1.0 | ×1.3 | ×1.7 | ×2.0 | ×4.0 | ×1.0 | ×1.0 | ×1.5 | ×2.0 | ×1.25 | ×1.5 | ×1.0 | ×1.0 |
| tCL [cycle (tCK)] | 16 | 16 | 16 | 10 | 10 | 16 | 16 | 16 | 16 | 13 | 13 | 10 | 10 |
| tRCD [cycle (tCK)] | 16 | 13 | 12 | 10 | 9 | 16 | 16 | 12 | 10 | 14 | 12 | 16 | 16 |
| tRP [cycle (tCK)] | 16 | 14 | 12 | 10 | 9 | 16 | 16 | 12 | 10 | 14 | 12 | 16 | 16 |
| tRAS [cycle (tCK)] | 38 | 34 | 28 | 24 | 20 | 38 | 38 | 31 | 24 | 35 | 31 | 38 | 38 |
| ACT+PRE E [nJ] | 11.8 | 10.4 | 8.7 | 7.4 | 6.3 | 11.8 | 11.8 | 9.3 | 7.4 | 10.6 | 9.3 | 11.8 | 11.8 |
| RD/WR E [nJ] | 13.5 | 13.6 | 13.7 | 11.1 | 11.2 | 13.6 | 13.7 | 13.6 | 13.7 | 12.3 | 12.4 | 11.1 | 11.2 |
| Per-rank standby P [W] | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 |

Table I: Timing parameters, energy, and power of the baseline DRAM, all region with high aspect ratio mat DRAM (All-HAR), CHARM, and SALAD organizations with 3% and 6% area overheads, where 1 tCK = 0.83ns.

When memory accesses are evenly spread across all the banks in a system and have low spatial locality, a memory access commonly results in a row buffer miss, which is only exacerbated by frequent DRAM refreshes. In this case, the average access latency (tAC) of SALAD over an entire DRAM die with 3% and 6% area overheads is 22.4ns and 21.2ns, respectively, whereas that of CHARM is 24.1ns and 23.9ns. For a given area overhead, SALAD outperforms CHARM in terms of average memory latency.

Figure 4 compares non-uniform access latencies (tAC) in both CHARM and SALAD according to the type of the region. CHARM is a two-level NUMA device with (a) slow edge regions and (e) fast center regions. In this setup, it is crucial to serve as many memory accesses as possible from the fast regions to maximize performance. If data pages are randomly distributed (i.e., page criticality-oblivious allocation), the chance for a memory request to fall onto a fast region is about 25%. By contrast, SALAD is a three-level NUMA device with (nearly) symmetric tAC's among the three regions (in the order of increasing distance from the I/O pads): (b) center regions, (c) edge regions, and (d) corner regions. Although slower than the center regions in CHARM, SALAD achieves lower tAC by 7% (11%) on average for a 3% (6%) area overhead. When hot pages exist in an application and the annotation/profiling techniques successfully map those pages to center DRAM regions, CHARM provides lower latency than SALAD for those pages. However, this latency gap shrinks if the application has ample spatial locality in memory accesses because both CHARM and SALAD have the same latency (tCL) for any row-buffer hit. SALAD provides lower latency to the remaining, less frequently accessed pages than CHARM, providing more robust performance without counting on the quality of hot page-aware allocation.

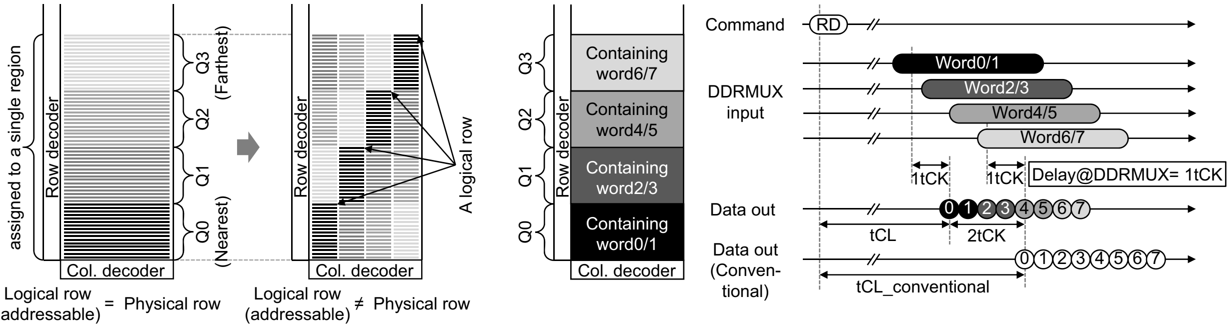
### C. Discussion

**Impact on memory controller design:** Because timing parameters vary depending on which region a page being accessed belongs to, a memory access scheduler must extract this information by parsing the row address field. A DRAM request that does not require data I/O, such as activate, precharge, and refresh, is free from DRAM I/O conflicts. Thus, region-aware command scheduling would suffice to ensure correctness without requiring any retries.

For refresh operations, a contemporary DRAM device automatically calculates the next target row address to refresh and generates an activate command. A memory controller is only required to send 8,192 refresh commands every 64ms interval, honoring the refresh timing constraints (e.g., refresh intervals time (tREFI) and refresh cycle time (tRFC)) [16], with no address information [2], [4]. Hence, no change is necessary for refresh operations on the memory controller.

**Read timing:** Similar to [38], [41], SALAD requires three types of RD commands with different tCL values for the center, edge, and corner regions. A regional group can accept RD commands with tCL values equal to or higher than its intrinsic tCL value, which is determined by the distance between the regions and the I/O pads of a device. When it receives a command with a latency higher than its intrinsic value, the command/address signals are latched at the command input buffers close to banks to delay the read process accordingly. Latching command/address values is more cost-effective than latching data values as the latter takes more bits (10 vs. 32 bits). This DRAM-side latching reduces the complexity of the DRAM access scheduler to support SALAD. If all regions have a single uniform tCL value (e.g., conventional DRAM), no conflict in data bus exists due to RD. With multiple tCL values, a RD scheduled at a certain cycle may occupy the data bus at the same time as a previously issued RD when the latter has higher tCL values. Using the *delayed* RD command, the memory controller can avoid this conflict without rescheduling the read operation at the following cycles.

**Write timing:** Writes necessitate a slightly different approach from reads. Because the data bus is bidirectional, the data receiver is not always on; instead, the receiver is turned on by the time when the actual data is placed on the bus (after tCWL). As tCWL varies depending on the regions being accessed, the memory controller must be aware of this parameter when scheduling writes.



(a) Changes in logical row composition for SOUP

(b) Intra-row pipelined access for SOUP

Figure 5: Design of SOUP (skewed organization of $\mu$banks with pipelined accesses): (a) changes in logical row composition and (b) intra-row pipelined access operations.

**Managing vendor-specific row address mapping:** Because the device row address mapping schemes vary according to DRAM vendors, the memory controller should know which bank/row addresses are mapped to the specific type of regions. One way to achieve this is to encode the row address mapping information on SPD (Serial Presence Detect) [15] within DIMM and do handshaking at booting.

## IV. SOUP: Exploiting Intra-Region Latency Asymmetry

There is a question that naturally arises for SALAD: what is the optimal number of the types of regions? Providing finer-granularity regions may further improve performance at the cost of complex hardware for both the DRAM device and memory controller. As an alternative, this section introduces SOUP (skewed organization of $\mu$banks with pipelined accesses), a low-cost microarchitectural technique to further reduce column access latency (tCL) uniformly with a negligible hardware cost. SOUP exploits serialization latency at I/O pins for data bursts, which offers timing slack for remote cells within a region to be read, while transferring data from local cells simultaneously during a column access.

### A. Motivation

In SALAD, a single bank is divided into two regions (half-banks) with different tCL's. This latency difference *across* regions originates from the different topological distance to I/O pads at the center of the chip. Likewise, *within* a region (half-bank), the intrinsic column access latency differs depending on the distance from the column decoder. Figure 5(a) shows a half-bank assigned to a SALAD region. If we divide this half bank into four subarray groups from Q0 (nearest) to Q3 (farthest) in the order of the proximity to the column decoder, the delay difference between Q0 and Q3 amounts to 1.8ns at 22nm technology according to our SPICE modeling (details in Section V). This delay is longer than 2 tCK in a DDR4-2400 system.

We can achieve a better-than-worst-case column access latency by exploiting serialization latency at I/O pads for *burst-oriented data transfer*, which is the norm in modern DRAM devices to exchange data with processors with limited pin count. For example, a 64B cache line is divided into 8 words (with 8 bytes per word), each being transferred at a rising or falling edge of the memory clock (DQS [16]). Thus, an entire cache line is transferred in four DQS cycles. The cache line delivered through wider inter-bank datalines (say, 32-bit width) in parallel is serialized using DDRMUXes. This serialization latency provides a timing slack for data from the farthest segment (Q3) to be read, while those from the nearest segment (Q0) are transferred through I/O pads.

### B. Design of SOUP

SOUP 1) partitions a single *logical* row into slices and interleaves them over the four latency segments (Q0 through Q3), and 2) starts a data burst for a `RD` command as soon as the data from the nearest segment (Q0) arrive. In this way SOUP effectively overlaps accesses to far segments with I/O transfers for near ones to save two tCK cycles for *all* regions.

**Modifications for the logical row composition:** SOUP decouples logical (addressable) rows from physical rows. Figure 5(a) compares the placement of a logical row in both a conventional DRAM device and SOUP. In both cases four logical rows are shown in different colors. In SOUP a single logical row is partitioned to four slices and interleaved over four segments (Q0 through Q3). Thus, once a `RD` command is issued, a quarter of column data coming from Q0 will arrive earlier than the other three quarters. The delay difference between Q0 and Q3 segments amounts to more than 2 tCK for DDR4-2400 devices at 22nm technology.

**Column access example with SOUP:** Suppose a cache line has eight words (Word0-Word7) with 8 bytes per word. As shown in Figure 5(b), SOUP places Word0/1, Word2/3, Word4/5, and Word7/8 to Q0, Q1, Q2, and Q3, respectively. Once a read command is issued, Word0/1 from the Q0 segment arrives at DDRMUX first, and then Word2/3, and so on. DDRMUX starts data transfers through I/O pins as soon as Word0/1 arrive, instead of waiting for the entire

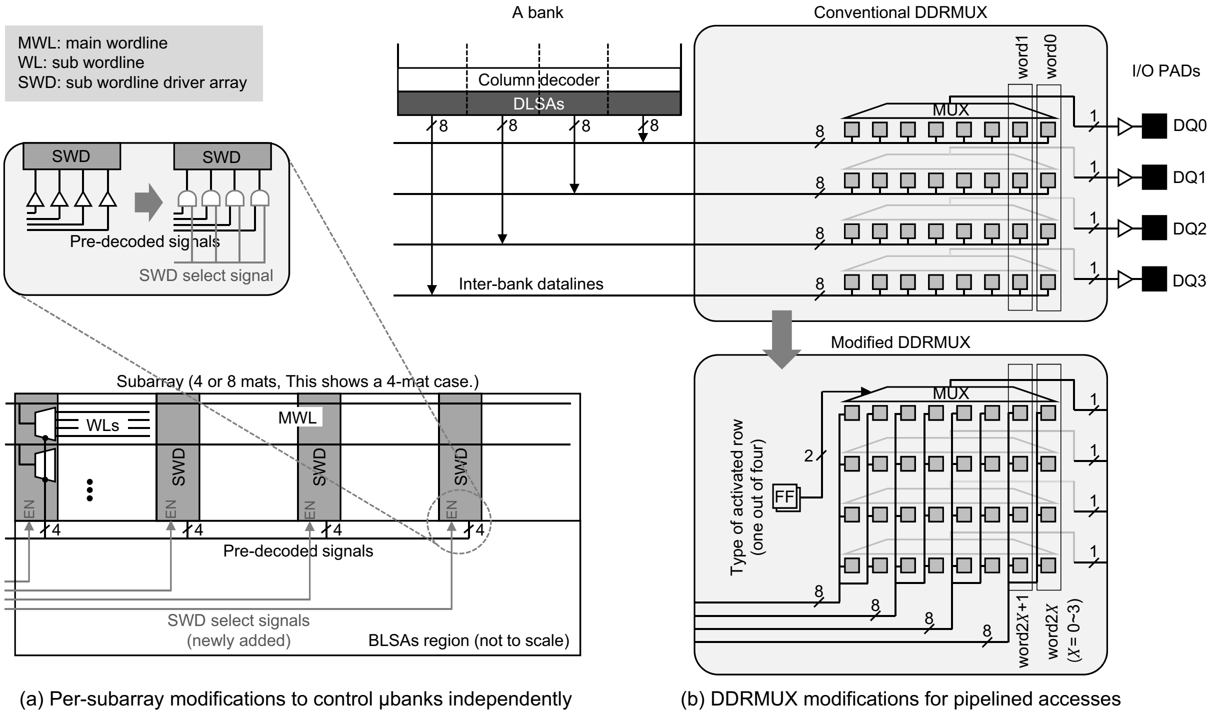

MWL: main wordline
WL: sub wordline
SWD: sub wordline driver array
SWD
SWD
Pre-decoded signals
SWD select signal
A bank
Column decoder
DLSAs
8
Inter-bank datalines
Conventional DDRMUX
word1
word0
MUX
I/O PADs
DQ0
DQ1
DQ2
DQ3
Modified DDRMUX
Type of activated row (one out of four)
FF
2
word2X+1
word2X (X= 0~3)
Subarray (4 or 8 mats, This shows a 4-mat case.)
WLs
MWL
SWD
EN
4
Pre-decoded signals
SWD select signals (newly added)
BLSAs region (not to scale)
(a) Per-subarray modifications to control µbanks independently
(b) DDRMUX modifications for pipelined accesses

Figure 6: Detailed implementation of SOUP. (a) Per-subarray modifications to control $\mu$banks independently. (b) DDRMUX modifications for pipelined accesses.

column data to arrive. Thus, tCL is determined by the delay of the Q0 segment (instead of the Q3 segment), allowing us to save two tCK cycles (1.7ns) in DDR4-2400 systems.

### C. Implementation of SOUP

To implement SOUP, DRAM should be able to diagonally activate four slices placed over four latency segments. To activate four subsets of mats within a subarray independently, we add four *sub-wordline driver (SWD) array select signals* to each subarray (see Figure 6(a)). More specifically, existing pre-decoded signals (for selecting one of the four WLs associated with a main WL) are AND-gated with a SWD select signal. The height of the subarray (along the Y-axis in Figure 6(a)) is commonly determined not by metal wires but by the size of sub-wordline drivers and that of the BLSAs. Therefore, the area overhead of adding four lines per subarray is negligible. By contrast, the drivers and receivers of those lines do incur area overhead. This overhead amounts to about 4K minimum-sized gates and 544 large-sized gates for the entire 8Gb DRAM die.

In conventional DRAMs, a large number of datalines are associated with a certain I/O pad. For pipelined access within a column access, those datalines should be transposed at DDRMUX to deliver the data words from the nearest segment first through I/O pins. Therefore, as Figure 6(b) shows, the direction of major metal lines in DDRMUX is changed (with no additional components introduced). There are four types of logical rows (each type is expressed as the same color in Figure 5(a)). Depending on which type of logical rows is activated, the position of the first word within a column changes. Therefore, *per-bank activated row type information* should be kept at an activation command (requiring 32 flip-flops and control circuits per DRAM die). Once a read (write) command is issued, the word number offset is determined using this information ($X$ in Figure 6(b) is determined). Still, there is no need to twist the connections between datelines and storage elements in DDRMUX. Instead, DDRMUX simply changes the order of transferring the latched words.

To estimate area overhead, we scale down 45nm NanGate open cell library [33] in reference to [23] and estimate the die size of 22nm 8Gb DRAM on the basis of [39]. According to our estimation, the area overhead is only 0.04% (0.033mm$^2$ over 82mm$^2$). The most pronounced energy overhead is due to switching four main WLs concurrently at a row activation. In our SPICE modeling/simulation, the row activation energy is increased by only 17pJ.

### D. Discussion

**Support for critical-word first:** JEDEC SDRAM standard [16] provides an optimization similar to *critical-word first* (CWF) to improve cache performance [11], [14]. At a read access the processor can specify the necessary word with the column address field. Then DRAM changes the intra-burst word order and transfers the critical word first. Figure 7(a) shows eight possible word-ordering patterns according to the critical word specified by the processor. Depending on the critical word specified, SOUP may not be able to deliver the critical word at the beginning of the data burst if it is mapped to the farthest segment, for example. However, with latest processors (e.g., Intel Haswell microarchitecture [9] or later), CWF means little, if any, because they have the 512-bit wide data path between L1 cache and L2 cache, which is equal to the burst size of a DRAM module. On the return path from the memory controllers to the core, the 64B data (a whole cache line) are transferred in parallel.

Nevertheless, we present a new alternative technique called Critical-Word Earliest Possible (CWEP) for those platforms that would benefit from CWF. With CWEP, the critical word access latencies of SOUP are always lower than or equal to those of the conventional design with CWF (see Figure 7(a)). Without CWEP (Figure 5(b)), tCL is determined by the delay of Word0 at DDRMUX, and the other words wait to be transferred. Thus, there is little room for advancing the turn of the rest words (Figure 7(b)). Instead, CWEP puts the turn of the critical word as earliest as possible although it may not be the first word being transferred (Figure 7(c)). With CWEP, even if the critical word is Word6 or Word7 (the worst case), the critical word access latency will be still equal to that of the conventional design with CWF (Figure 7(a)). If the critical word is one of the other words, SOUP with CWEP has a lower access latency for the critical word, which is achieved by either of the two possible word-ordering patterns (one for even critical words and the other for odd critical words).

Finally, a previous study [5] shows that the critical words of most memory accesses in common applications are Word0 (67% on average). Thus, the critical word support is likely less of an issue for SOUP even without CWEP support.

**Other target devices and alternative designs:** SOUP can also be applied to conventional DRAMs; therefore, it can be regarded as an orthogonal technique to SALAD architecture. For conventional DRAMs, with SOUP, tCL is reduced by 3tCK (the in-bank asymmetry in column access latencies is as large as 3.6ns, but the tCL reduction is limited by the burst length). As an alternative design instead of using the distributed sliced rows, we can utilize the latency skew in the column decoder (in the row direction). However, in this case, tCL is reduced by only 1tCK.

**Repairing faulty cells in SOUP-N-SALAD:** For SOUP-N-SALAD, faulty cells should be replaced with only spares within the same quarter of the half-bank. Though commercial DRAM designs commonly replace faulty cells with spares within the same subarray, SOUP-N-SALAD, like other fine-grained architectures [24], limits the flexibility of using spare rows by disallowing a faulty row in one subarray from being replaced by a spare row in a different subarray.

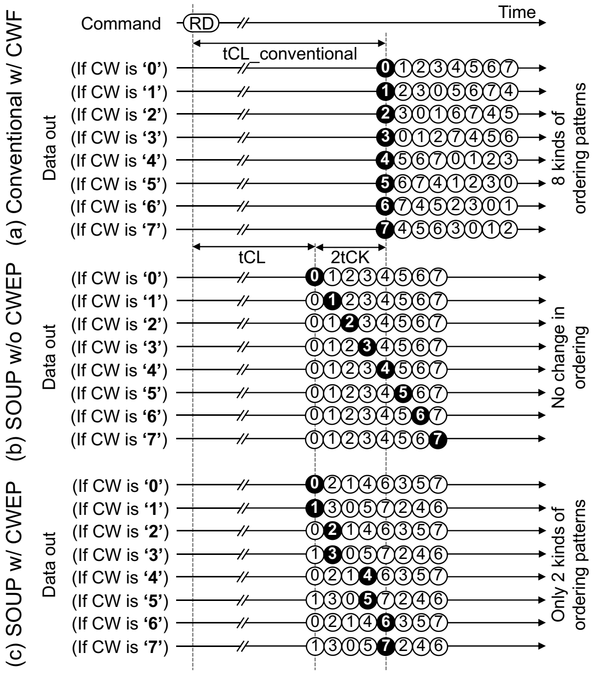

(a) Conventional w/ CWF
Command
RD
Time
tCL_conventional
Data out
(If CW is '0') 0 1 2 3 4 5 6 7
(If CW is '1') 1 2 3 0 5 6 7 4
(If CW is '2') 2 3 0 1 6 7 4 5
(If CW is '3') 3 0 1 2 7 4 5 6
(If CW is '4') 4 5 6 7 0 1 2 3
(If CW is '5') 5 6 7 4 1 2 3 0
(If CW is '6') 6 7 4 5 2 3 0 1
(If CW is '7') 7 4 5 6 3 0 1 2
8 kinds of ordering patterns
tCL
2tCK
(b) SOUP w/o CWEP
Data out
(If CW is '0') 0 1 2 3 4 5 6 7
(If CW is '1') 0 1 2 3 4 5 6 7
(If CW is '2') 0 1 2 3 4 5 6 7
(If CW is '3') 0 1 2 3 4 5 6 7
(If CW is '4') 0 1 2 3 4 5 6 7
(If CW is '5') 0 1 2 3 4 5 6 7
(If CW is '6') 0 1 2 3 4 5 6 7
(If CW is '7') 0 1 2 3 4 5 6 7
No change in ordering
(c) SOUP w/ CWEP
Data out
(If CW is '0') 0 2 1 4 6 3 5 7
(If CW is '1') 1 3 0 5 7 2 4 6
(If CW is '2') 0 2 1 4 6 3 5 7
(If CW is '3') 1 3 0 5 7 2 4 6
(If CW is '4') 0 2 1 4 6 3 5 7
(If CW is '5') 1 3 0 5 7 2 4 6
(If CW is '6') 0 2 1 4 6 3 5 7
(If CW is '7') 1 3 0 5 7 2 4 6
Only 2 kinds of ordering patterns

Figure 7: Comparison between critical word first (CWF) and critical word earliest possible (CWEP): (a) conventional DRAM with CWF, (b) SOUP without CWEP, and (c) SOUP with CWEP.

**Applying to new memory types:** Recently, a lot of 3D-stacked memories are proposed/launched, such as HBM [17] and HMC [36]. SOUP-N-SALAD can also be applied to these memories and improve latency. Even if 3D-stacked memory stacks die in a vertical direction, there is still a difference in horizontal distance between DRAM cells and the data path. Moreover, the idea of SOUP-N-SALAD can be applied to the non-uniform distance across multiple dies in a vertical direction.

## V. Experimental Methodology

We model a chip-multiprocessor system to evaluate the performance and energy efficiency of SOUP-N-SALAD. The system consists of 16 out-of-order cores and 4 memory channels, where a core operates at 3.6GHz, issues/commits up to 4 instructions per cycle, and has 64 reorder buffers. Each core is connected to separate L1-I and L1-D caches and a unified L2 cache, all with 64B cache lines and 4 banks. Also, a shared L3 last level cache consists of 16 cache slices, where one slice is connected to one L2 cache. The capacity and set associativity of an L1, L2, and L3 cache are (32KB, 4), (256KB, 8) and (1MB/slice, 16), respectively. A MOESI protocol and a linear hardware prefetcher [19] that detects and prefetches streams of consecutive memory accesses are used. Each memory channel consists of 4 ranks of DDR4-2400 modules, where each rank consists of 18 8Gb ×4

| Resource | Value |
|---|---|
| Number of cores, MCs | 16 cores, 4 MCs |
| Coherence policy | MOESI |
| Per core: | |
| Freq, issue/commit width | 3.6 GHz, 4/4 slots |
| Issue policy | Out-of-Order |
| L1 I/D $ type/size/associativity | Private/32KB/4 |
| L2 $ type/size/associativity | Private/256KB/8 |
| L1, L2 $ line size | 64B |
| Hardware (linear) prefetch | On |
| L3 $ type/size/associativity | Shared/16MB/16 |
| Per memory controller (MC): | |
| # of channels, Request Q size | 1 Ch, 32 entries |
| Capacity and bandwidth per rank | 16GB, 19.2GB/s |
| Scheduling policy | PAR-BS [32] |
| DRAM page policy | Adaptive open [13] |

Table II: Default parameters of the simulated system.

DRAM devices (2 for ECC). The memory controller (MC) of each channel has 64 request queue entries and adopts the parallelism-aware batch scheduling [32] and adaptive open page management policy [13] by default. To support the multi-level latency values of SALAD/CHARM, the MC initiates different counter values depending on the region that an ACT/PRE command falls on. It schedules a RD/WR command with the minimal available latency that satisfies the minimal tCL of the corresponding region and avoids conflicts in the data I/O by the *delayed* read commands. We modify McPAT [26] for CPU modeling at the 14nm technology node and McSimA+ [1] for performance simulation.

SPEC CPU2006 [12] benchmark suite is used for single- and multi-programmed workloads. We identify and use the most representative simulation point of each application using Simpoint [37], which consists of 100M instructions. We measure the memory accesses per kilo-instructions (MAPKI) for the SPEC applications to classify the nine most memory-intensive applications as spec-high: mcf, milc, leslie3d, soplex, GemsFDTD, libquantum, lbm, sphinx3, and omnetpp. We use two multi-programmed workloads called mix-high and mix-blend, where the former consists of two instances of the spec-high applications and the latter of an instance of perlbench, bzip2, cactusADM, dealII, libquantum, zeusmp, soplex, h264ref, mcf, xalancbmk, hmmer, milc, calculix, lbm, bwaves, and wrf. MICA [27], [28] (a key-value store), PARSEC [3], and SPLASH-2 [45] are used for multi-threaded workloads.

We use the latency and energy values of the baseline DDR4, All-HAR, CHARM, and SALAD devices obtained from SPICE simulation in Table I. A modified PTM low-power model [48] assuming a 22nm, 3-metal layer process is used. We size transistors to satisfy the baseline DDR4-2400 timing specifications [16]. We set target area overheads over the baseline as 3% and 6%, which are the representative design points with high performance gains in CHARM [41] for fair comparison. All-HAR increases the aspect ratio (AR) of all the mats in a DRAM device, similar to RLDRAM [5], with ARs of ×1.3 and ×1.7, which correspond to overall area overheads of 3% and 6%, respectively. Increasing the AR of the mats at the 8 center banks out of 16 banks per device by 2 and 4 times leads to 3% and 6% area overheads, which is consistent with the results in [41]. For SALAD, regions at the (center, edge, corner) groups have the mats with aspect ratios of (×1.0, ×1.25, ×1.5) and (×1.0, ×1.5, ×2.0) to have 3% and 6% area overheads, respectively.

## VI. Evaluation

We quantify the system-level performance and energy efficiency benefits of SOUP-N-SALAD over previous low-latency DRAM architectures using various workloads.

### A. Performance and Energy Efficiency

**Results of SALAD:** Figure 8 shows the performance (IPC) and system-level energy efficiency (MIPS$^2$/W) of the three DRAM organizations–HAR on all mats (HAR), CHARM (CH), and SALAD (SA) with 3% and 6% area overheads over the baseline DDR4-2400 device, using single-threaded and multi-programmed SPEC CPU2006 [12] workloads, and multi-threaded MICA [27], [28], SPLASH2 [45], and PARSEC [3] applications. We apply different page mapping schemes for SALAD and CHARM. First, SALAD and CHARM-O adopt a hot page-oblivious mapping without any additional software or hardware modification. Second, to obtain the results of CHARM-A (CHARM with hot page-aware mapping), we employ an optimal page allocation policy based on offline profiling to allocate frequently accessed (hot) pages to fast regions. We present the aggregate IPC values for multi-programmed workloads. For single-threaded workloads, only one memory channel is populated to stress the main memory system.

We make the following key observations. First, SALAD with hot page-oblivious mapping achieves performance and EDP (an inverse of MIPS$^2$/W) comparable to, or better than, CHARM-A. With 3% (6%) area overhead, both SALAD and CHARM-A improve IPC by 8.5% (10.2%) and 8.4% (9.2%) on average for spec-high applications. Moreover, the energy efficiency (MIPS$^2$/W) is improved over the baseline DDR4 configuration by 16.8% (20.4%) and 17.2% (12.6%), respectively. CHARM-A is effective for applications with uneven memory access frequency over pages (e.g., 429.mcf), but not for applications with evenly distributed access patterns (e.g., 470.lbm and 462.libquantum). By contrast, SALAD is appealing as it provides robust performance even without careful, criticality-aware page allocation, whereas CHARM-O improves IPC of spec-high applications only by 3.6% on average. Similar to the baseline and All-HAR, SALAD has the access latency (tAC) mostly the same over an entire device, whereas CHARM has a huge difference in tAC between the center and edge/corner regions.

Second, SALAD is more effective for multi-programmed and multi-threaded workloads than single-threaded work-

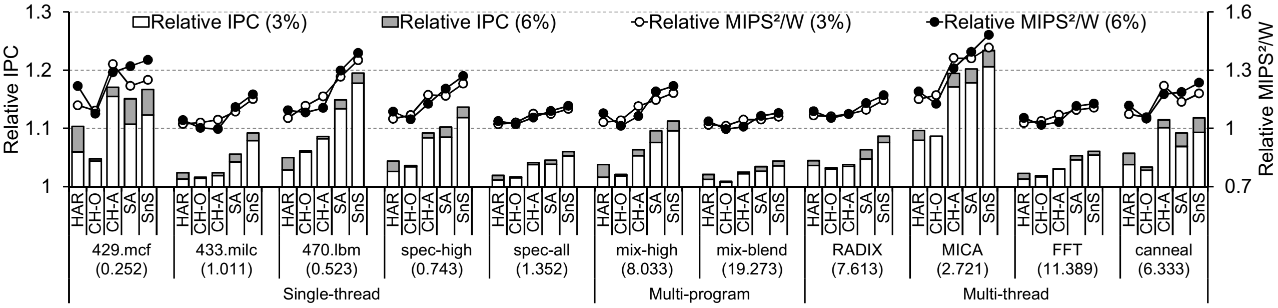


Figure 8: The relative performance (IPC) and MIPS$^2$/W of DRAM organizations with HAR on all mats (`HAR`), CHARM (`CH`), SALAD (`SA`), and SOUP-N-SALAD (`SnS`) with 3% and 6% area penalty compared to the baseline DDR4-2400 device on single-threaded and multi-programmed SPEC CPU2006, multi-threaded MICA, SPLASH2, and PARSEC applications. The number within the round brackets in the label per application is the absolute IPC of the baseline.

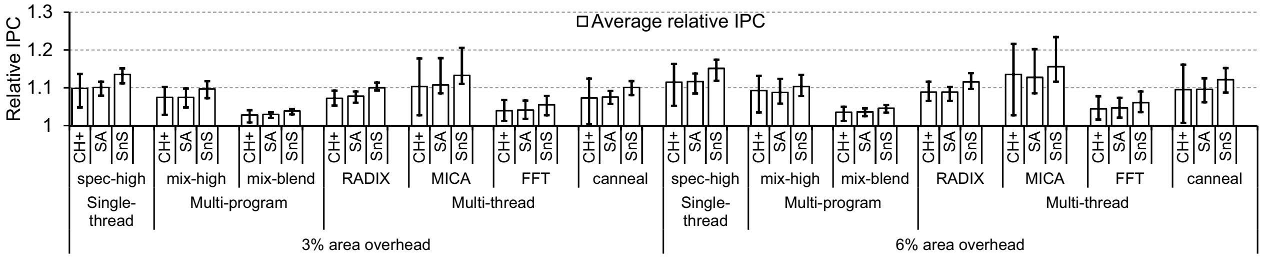


Figure 9: The relative performance (IPC) of DRAM organizations with various cases of hot page allocation on CHARM+ (`CH+`), SALAD (`SA`) and SOUP-N-SALAD (`SnS`) with 3% and 6% area overhead.

loads. Unlike single-threaded workloads, multi-programmed and multi-threaded workloads execute multiple threads simultaneously, which have widely varying characteristics in memory access pattern, frequency, and footprint. This incurs more complex problems such as bank-level interference [20], [46] with an increasing core count because main memory is one of the most important shared resources in modern computer systems. Especially, a multi-threaded workload often divides a large array into similarly sized chunks and executes each task at a similar pace with evenly distributed access patterns. Thus, SALAD is more efficient than CHARM in that the tAC has uniformly low access latency over the whole device, leading to a better load balance, whereas CHARM improves only a part of the device. SALAD outperforms CHARM-A in most multi-threaded and multi-programmed workloads, except for a special case such as canneal which frequently accesses a small number of pages. SALAD improves IPC of mix-high and mix-blend workloads by 7.6% and 2.6% with 3% area overhead and by 9.6% and 3.4% with 6% area overhead over the baseline. It also achieves 9.5%, 9.5%, 36.0%, and 13.6% better MIPS$^2$/W over the baseline DDR4 on RADIX (SPLASH-2), FFT (SPLASH-2), MICA, and canneal (PARSEC).

Finally, increasing the aspect ratio of all the mats of a device (All-HAR) is not as effective as SALAD. With 3% area overhead, All-HAR improves IPC of spec-high and spec-all (the average over all SPEC CPU2006 applications) only by 2.6% and 1.1%, respectively. This is because All-HAR has a single tCL, which is determined by the topological distance between I/O pads and the farthest region of the device. Thus, SALAD is more effective than All-HAR due to its better-than-worst-case design at a given area constraint.

**Results of SOUP-N-SALAD:** Figure 8 shows the relative performance (IPC) and system-level energy efficiency (MIPS$^2$/W) of SOUP-N-SALAD (`SnS`) using single-threaded, multi-programmed, and multi-threaded workloads as before. SOUP makes tCLs shorter than the default SALAD by two memory clock cycles (tCK). Combined with SOUP, SOUP-N-SALAD achieves performance comparable to or better than SALAD. Without additional area overhead, employing SOUP improves the IPC of SALAD by 3.3% (3.4%), 2.0% (1.7%) and its MIPS$^2$/W by 6.3% (6.5%), 3.5% (2.9%) for spec-high applications and mix-high workload, with 3% (6%) area overhead. Also, SOUP-N-SALAD achieves 2.8% (2.3%), 2.8% (3.2%), 0.8% (0.8%), 2.4% (2.6%) IPC and 5.0% (4.0%), 5.6% (8.8%), 1.2% (1.3%), 4.4% (4.8%) MIPS$^2$/W improvements over SALAD for RADIX, MICA, FFT, and canneal, respectively.

### *B. Performance Robustness*

In this section we quantify the performance robustness of SOUP-N-SALAD by measuring IPC variation with 7 different page allocation policies. Because there are three regions in SALAD, there are 6 combinations to allocate hot/warm/cold pages to the three different regions. Additionally, we also use a criticality-oblivious memory allocation policy, to evaluate 7 allocation policies in total. For fair comparison, we extend the original CHARM to have three latency regions (instead of two) and call it CHARM+ (CH+), through which SALAD and CHARM+ have comparable average access time (tAC) for a given workload. The main difference between CHARM+ and SALAD is the (a)symmetry of latency across regions.

Figure 9 shows the IPC variations of CHARM+, SALAD, and SOUP-N-SALAD using error bars. The results demonstrate SALAD and SOUP-N-SALAD achieve significantly lower IPC variations compared to CHARM+. More specifically, with 3% (6%) area overhead, CHARM+ has 8.9% (11.1%) and 7.4% (9.6%) IPC variations, whereas SALAD has 3.7% (5.3%), 5.0% (6.6%) IPC variations and SOUP-N-SALAD 3.9% (5.6%), 4.5% (5.6%) variations for spec-high and mix-high, respectively. The results demonstrate that CHARM+ is much more sensitive to memory allocation policies than SALAD and SOUP-N-SALAD. As stated in Section III, criticality-aware allocation can be expensive, fragile, or both, so achieving robust performance without requiring criticality-aware memory allocation is appealing. This performance robustness is likely to be more pronounced in the future with an ever-increasing number of concurrent threads contenting for memory space and bandwidth.

## VII. Related Work

There are various proposals to enhance performance and energy efficiency of the main memory system, which is one of the most important shared resources in contemporary computer systems. We classify them by subject.

**Low latency DRAM with fewer cells per bitline:** Reduced Latency DRAM [5], [14] and FCRAM [8] use fewer cells per bitline than the conventional DRAM device to reduce memory access latency. However, these DRAM organizations have a huge area overhead with higher cost per bit, which prevents their widespread adoption to replace conventional DRAM devices. By contrast, to reduce high area cost, Lee et al. [25] propose the Tiered-Latency DRAM architecture by using an isolation transistor to split each long bitline into fast and slow regions. It improves the memory performance by shortening several timing parameters such as tRCD and tRP. However, it is not sufficient to optimize DRAM access latency, as tCL is also a dominant factor with tRCD and tRP. Similar to the Tiered-Latency DRAM, CHARM [41] proposes a technique to reduce tAC (tRCD + tCL) with short tCL of the center region; but CHARM has both slow and fast regions in a DRAM device, necessitating careful page allocation. With our proposals which extended [40], tAC is (nearly) the same over an entire device, while providing robust performance without requiring a criticality-aware allocation scheme.

**Reducing DRAM latency using timing margin of DRAM:** ChargeCache [10] and NUAT [38] reduce memory access latency by exploiting the timing margin of DRAM parameters. The timing parameters for sensing/restore (tRAS/tRCD) assume that a cell is not fully charged due to leakage. The time required for sensing/restore is shorter when a cell is fully charged, right after refresh or row access. The system performance can be improved with minimizing the timing margin for these cases. However, this approach has limitation on improving performance as it can reduce only tRCD and tRAS, not tCL which has greater impact on performance. Row-buffer decoupling [34] hides precharge latency on most cases by decoupling row buffers from bitlines, which is orthogonal to SOUP-N-SALAD.

**Subarray parallelism:** Microbank [42], SALP [24], Half-DRAM [47], and SSA [43] improve performance by exploiting DRAM subarray characteristics. They maximize bank-level parallelism by populating logical banks that can operate independently. Furthermore, the logical banks also help DRAM refresh operations by activating several rows in a bank simultaneously. However, there is still a room for further performance improvement on latency-sensitive workloads, even though they show significant improvement in memory-intensive workloads.

**3D-stacked DRAM Architectures:** There have been numerous proposals [6], [18], [29], [30], [44] on 3D-stacked DRAM architectures to improve memory bandwidth and capacity while adopting through-silicon vias (TSVs) to stack DRAM dies on top of a processor die. TSV-RDIMM [35] is a state-of-the-art technology, which stacks multiple DDR4 dies in a single processor package. It is also possible to apply them to our proposed architectures to gain additional performance benefits.

## VIII. Conclusion

We have proposed two microarchitectural techniques to provide uniformly low access time (tAC) over the entire DRAM device: SALAD and SOUP. SALAD reduces the access latency of all the banks in a DRAM device by 1) allowing each bank to have different tCL based on its distance to the I/O pads, and 2) reducing the activate and precharge time of the remote banks by decreasing the bitline capacitance of the mats for those banks. This is different from CHARM [41], which applies the aforementioned techniques only to the center banks, thus yielding non-uniform tAC among banks. SALAD achieves even lower access latency than a DRAM architecture that applies HAR mats to the entire DRAM device (All-HAR) at the same area budget. The second technique, SOUP, leverages asymmetry on column access latencies *within* a DRAM region. SOUP exploits intra-DRAM-bank latencies with fine-grained access latency partitioning, which saves 2tCK in column access latencies (tCL) for all regions. The resulting design, called *SOUP-N-SALAD*, improves both IPC and EDP by 9.6% (11.2%) and 18.2% (21.8%) over the baseline DDR4 device, respectively, for memory-intensive SPEC CPU2006 workloads without any software modifications, while incurring only 3% (6%) area overhead.

## Acknowledgments

This research was supported in part by the Future Semiconductor Device Technology Development Program funded by MOTIE and KRSC (10044735), Next-Generation Informaion Computing Development Program through NRF funded by MSIP (2015M3C4A7065647), and Basic Science and Research Program (2014R1A1A1005894) through NRF.

## References

[1] J. Ahn, S. Li, S. O, and N. P. Jouppi, "McSimA+: A Manycore Simulator with Application-level+ Simulation and Detailed Microarchitecture Modeling," in *ISPASS*, 2013.

[2] I. Bhati, M.-T. Chang, Z. Chishti, S.-L. Lu, and B. Jacob, "DRAM Refresh Mechanisms, Penalties, and Trade-Offs," *TC, IEEE*, 2016.

[3] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *PACT*, 2008.

[4] K. K.-W. Chang *et al.*, "Improving DRAM performance by parallelizing refreshes with accesses," in *HPCA*, 2014.

[5] N. Chatterjee *et al.*, "Leveraging Heterogeneity in DRAM Main Memories to Accelerate Critical Word Access," in *MICRO*, 2012.

[6] Y.-J. Chen, C.-L. Yang, and J.-J. Chen, "Distributed Memory Interface Synthesis for Network-on-Chips with 3D-Stacked DRAMs," in *ICCAD*, 2012.

[7] J. Choi *et al.*, "Multiple Clone Row DRAM: A Low Latency and Area Optimized DRAM," in *ISCA*, 2015.

[8] Fujitsu, "FCRAM," http://www.fujitsu.com/global/products/devices/semiconductor/memory/fcram/.

[9] P. Hammarlund *et al.*, "Haswell: The fourth-generation intel core processor," *Micro, IEEE*, no. 2, pp. 6–20, 2014.

[10] H. Hassan *et al.*, "ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality," in *HPCA*, 2016.

[11] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. Morgan Kaufmann Publishers Inc, 2012.

[12] J. L. Henning, "SPEC CPU2006 Memory Footprint," *Computer Architecture News*, vol. 35, no. 1, 2007.

[13] Intel, *Intel Xeon Processor 7500 Series Datasheet*, 2010.

[14] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann Publishers Inc, 2007.

[15] JEDEC, "SPD General Standard," 2008.

[16] JEDEC, "DDR4 SDRAM Specification," 2012.

[17] JEDEC, "High Bandwidth Memory (HBM) DRAM," 2013.

[18] D. Jevdjic, S. Volos, and B. Falsafi, "Die-Stacked DRAM Caches for Servers," in *ISCA*, 2013.

[19] N. P. Jouppi, "Improving Direct-mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers," in *ISCA*, 1990.

[20] D. Jung, S. Li, and J. Ahn, "Large Pages on Steroids: Small Ideas to Accelerate Big Memory Applications," *IEEE CAL*, vol. PP, no. 99, pp. 1–1, 2015.

[21] B. Keeth, R. J. Baker, B. Johnson, and F. Lin, *DRAM Circuit Design*, 2nd ed. IEEE, 2008.

[22] C. Kim, D. Burger, and S. W. Keckler, "An Adaptive, Non-uniform Cache Structure for Wire-delay Dominated On-chip Caches," in *ASPLOS*, 2002.

[23] D. H. Kim and S. K. Lim, "Design Quality Trade-Off Studies for 3-D ICs Built With Sub-Micron TSVs and Future Devices," 2012.

[24] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA*, 2012.

[25] D. Lee *et al.*, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," in *HPCA*, 2013.

[26] S. Li *et al.*, "The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing," *ACM TACO*, vol. 10, no. 1, 2013.

[27] S. Li *et al.*, "Architecting to Achieve a Billion Requests Per Second Throughput on a Single Key-Value Store Server Platform," in *ISCA*, 2015.

[28] H. Lim, D. Han, D. G. Andersen, and M. Kaminsky, "MICA: A Holistic Approach to Fast In-Memory Key-Value Storage," in *NSDI*, 2014.

[29] G. H. Loh and M. D. Hill, "Efficiently Enabling Conventional Block Sizes for Very Large Die-stacked DRAM Caches," in *MICRO*, 2011.

[30] N. Madan *et al.*, "Optimizing Communication and Capacity in a 3D Stacked Reconfigurable Cache Hierarchy," in *HPCA*, 2009.

[31] Micron, "Micron 8Gb x4 DDR4 SDRAM datasheet," https://www.micron.com/products/dram/ddr4-sdram/#8Gb.

[32] O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.

[33] Nangate, "Nangate Open Cell Library," http://www.nangate.com/?page_id=22.

[34] S. O, Y. H. Son, N. S. Kim, and J. Ahn, "Row-Buffer Decoupling: A Case for Low-Latency DRAM Microarchitecture," in *ISCA*, 2014.

[35] R. Oh *et al.*, "Design technologies for a 1.2 V 2.4 Gb/s/pin high capacity DDR4 SDRAM with TSVs," in *VLSI Circuits Digest of Technical Papers, Symposium on*, 2014.

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

[37] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior," in *ASPLOS*, 2002.

[38] W. Shin, J. Yang, J. Choi, and L.-S. Kim, "NUAT: A Non-uniform Access Time Memory Controller," in *HPCA*, 2014.

[39] 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 *ISSCC*, 2012.

[40] Y. H. Son, H. Cho, Y. Ro, J. W. Lee, and J. Ahn, "SALAD: Achieving Symmetric Access Latency with Asymmetric DRAM Architecture," *IEEE CAL*, vol. PP, no. 99, 2016.

[41] Y. H. Son, S. O, Y. Ro, J. W. Lee, and J. Ahn, "Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations," in *ISCA*, 2013.

[42] Y. H. Son *et al.*, "Microbank: Architecting Through-Silicon Interposer-Based Main Memory Systems," in *SC*, 2014.

[43] A. N. Udipi *et al.*, "Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores," in *ISCA*, 2010.

[44] D. H. Woo, N. H. Seong, D. L. Lewis, and H. H. S. Lee, "An Optimized 3D Stacked Memory Architecture by Exploiting Excessive, High-Density TSV Bandwidth," in *HPCA*, 2010.

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

[46] H. Yun, R. Mancuso, Z. P. Wu, and R. Pellizzoni, "PALLOC: DRAM Bank-aware Memory Allocator for Performance Isolation on Multicore Platforms," in *RTAS*, 2014.

[47] T. Zhang *et al.*, "Half-DRAM: a High-bandwidth and Low-power DRAM Architecture from the Rethinking of Fine-grained Activation," in *ISCA*, 2014.

[48] W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for Sub-45nm Design Exploration," in *ISQED*, 2006.