Heterogeneous HMC+DDRx Memory Management for Performance-Temperature Tradeoffs

MOHAMMAD HOSSEIN HAJKAZEMI and MOHAMMAD KHAVARI TAVANA, George Mason University
TINOOSH MOHSENIN, University of Maryland Baltimore County
HOUMAN HOMAYOUN, George Mason University

Q1 3D-DRAMs are emerging as a promising solution to address the memory wall problem in computer systems. However, high fabrication cost per bit and thermal issues are the main reasons that prevent architects from using 3D-DRAM alone as the main memory building block. In this article, we address this issue by proposing a heterogeneous memory system that combines a DDRx DRAM with an emerging 3D hybrid memory cube (HMC) technology. Bandwidth and temperature management are the challenging issues for this heterogeneous memory architecture. To address these challenges, first we introduce a memory page allocation policy for the heterogeneous memory system to maximize performance. Then, using the proposed policy, we introduce a temperature-aware algorithm that dynamically distributes the requested bandwidth between HMC and DDRx DRAM to reduce the thermal hotspot while maintaining high performance. We take into account the impact of both core count and HMC channel count on performance while using the proposed policies. The results show that the proposed memory page allocation policy can utilize the memory bandwidth close to 99% of the ideal bandwidth utilization. Moreover, our temperate-aware bandwidth adaptation reduces the average steady-state temperature of the HMC hotspot across various workloads by 4.5 K while incurring 2.5% performance overhead.

CCS Concepts: • **Computer systems organization** → **Heterogeneous (hybrid) systems**; • **Hardware** → **Memory and dense storage**; *Dynamic memory*;

Additional Key Words and Phrases: Heterogeneous memory, bandwidth, hybrid memory cube, temperature

ACM Reference format:
Mohammad Hossein Hajkazemi, Mohammad Khavari Tavana, Tinoosh Mohsenin, and Houman Homayoun. 2017. Heterogeneous HMC+DDRx Memory Management for Performance-Temperature Tradeoffs. *J. Emerg. Technol. Comput. Syst.* 14, 1, Article 4 (July 2017), 21 pages.
https://doi.org/10.1145/3106233

1 INTRODUCTION

As Moore's Law continues to drive technology scaling down to the nanometer realm, main memory DRAM scaling faces some serious challenges in capacity, speed, bandwidth, and power. In particular, pin count constraint is one of the major issues in scaling conventional DRAMs including

Q2 Author's addresses: M. H. Hajkazemi, M. Khavari-Tavana, and H. Homayoun, Electrical and Computer Engineering Department, George Mason University; T. Mohsenin, Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County.

https://doi.org/10.1145/3106233

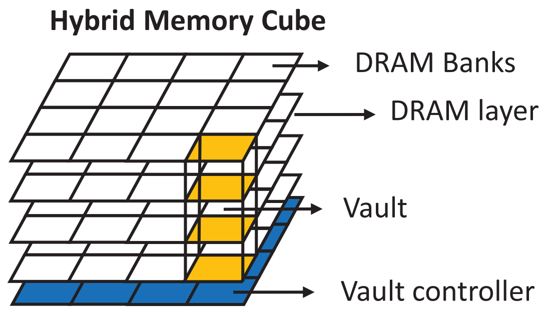


Fig. 1. HMC organization.

DDRx technology and results in performance degradation of the entire computing system (Zhang et al. 2014). Recent work has shown that the application bandwidth requirement in both medium-end and high-end processors is increasing rapidly (Greenberg 2012; Atwood 2011). Therefore, the pin constraint can create a bottleneck for high bandwidth-demanding applications, and exacerbates the memory wall challenge.

Three-dimensional (3D) integration is a key enabler to address this problem by using through-silicon vias (TSVs) (Meng et al. 2012; Jeddeloh and Keeth 2012; Tajik et al. 2013; Homayoun et al. 2012). With 3D integration, different layers of dies are stacked using fast interconnects (TSV) with a latency as low as few picoseconds. TSVs improve the capacitance per connection and thus reduce connection power dissipation by as much as 6 times (Bansal 2011). Moreover, they cut back the connection length by 200 times (Bansal 2011). Furthermore, TSVs provide higher number of connections, leading to higher throughput.

By exploiting 3D integration, we are able to stack multiple layers of DRAM, resulting in shorter memory access latency to potentially address the memory wall problem. Stacking DRAM gives us the opportunity to have parallel accesses to DRAM banks, which results in higher maximum achievable bandwidth.

Compared to the conventional DRAM architecture (2D), 3D-DRAM results in better performance by offering higher bandwith and lower latency. Hybrid memory cube (HMC) is an emerging 3D memory interface and design introduced by Micron to address the inefficiency of DDRx DRAMs (Jeddeloh and Keeth 2012). Figure 1 shows the HMC organization.

As Figure 1 presents, HMC stacks up to eight layers of standard DRAM building blocks on a logic layer. Each DRAM layer is segmented into multiple partitions composed of two banks. Adjacent vertical partitions constitute vaults that are controlled by vault controllers residing on the logic layer (Jeddeloh and Keeth 2012). This helps in simplifying the memory controller on the processor end. The processor's memory controller needs to send higher-level commands, that is, only read and write commands (Ahn et al. 2015a), without being concerned about timing and scheduling. However, 3D integration used in HMC imposes a significant power density challenge, as highlighted in a 2013 report by Rambus (Ming 2013). Higher power density causes many temperature-related problems, including extra cooling costs, reliability, wear-out, and leakage power issues (Kang et al. 2014). For example, having a higher number of stacked layers increases the heat resistivity of the entire chip package, which results in both higher peak and steady-state temperature. It also complicates the chip packaging process that makes the design more vulnerable to various failure mechanisms (Srinivasan et al. 2005).

Besides the thermal issues, fabrication cost is another challenge that limits the application of HMC. As the capacity of each HMC cube is limited to 2–4GB (Consortium 2014), several cubes need to be chained together to build the larger capacity required. This is not a practical option in terms of cost as well as design feasibility. Therefore, conventional 2D, DDRx DRAM, is indispensable to maintain the high capacity requirement of DRAM to achieve high performance and avoid the thermal and cost challenges associated with the new 3D technology.

A heterogeneous memory system that combines 2D- and 3D-DRAM can simultaneously exploit the high capacity, low cost, and low thermal footprint of 2D and high bandwidth and low access latency of 3D. However, the challenge is managing the two substantially different designs effectively to exploit their benefits simultaneously. In our earlier work (Tran et al. 2013), we attempted to address this issue; however, that work does not model HMC, and, instead, it studies a generic Q3 3D-stacked DRAM. Moreover, despite proposing a new policy to achieve higher QoS, we did not address the thermal challenge of 3D memory (Tran et al. 2013).

In this article, we introduce a heterogeneous HMC+DDRx memory system. The focus of this article is to address both performance and temperature challenges associated with the proposed memory architecture, simultaneously, by introducing performance-temperature-aware memory management mechanisms. Over-utilization of either HMC or DDRx DRAM results in bandwidth congestion and incurs a large performance loss. Furthermore, utilizing HMC to maximize the performance benefits can lead to thermal hotspots, which in turn can severely affect performance, due to thermal emergency response such as throttling. In order to utilize both HMC and DDRx DRAM efficiently, our memory management mechanism allocates the memory pages in an interleaved manner considering the system temperature and performance.

To the best of our knowledge, this is the first article to simultaneously address the performance and temperature challenges in a heterogeneous HMC+DDRx DRAM memory subsystem. The main contributions of this work are as follows:

- We show that a heterogeneous HMC+DDRx is an alternative for conventional DDRx and plain HMC memory system, which addresses the performance challenge and thermal issues of 3D integration, while achieving high performance.
- We show that in heterogeneous DDRx+HMC, the average memory access latency changes substantially across various bandwidth allocation and therefore suggests the need for a bandwidth-aware allocation policy to minimize the latency. We propose a runtime memory page allocation policy to efficiently utilize the bandwidth.
- We introduce a dynamic temperature-aware policy that utilizes our proposed heterogeneous DRAM based on the operating temperature of the HMC and the current phase of the workload. As a result, by allocating bandwidth to the HMC and the DDRx DRAM dynamically, we reduce the steady-state temperature.
- We perform a sensitivity analysis on the proposed bandwidth allocation policy to study how various request distribution with the same ratio can affect the accuracy of the proposed policy.
- We study a diverse range of workloads and architectures to analyze the benefits of the proposed heterogeneous memory performance in future architectures.
- We investigate how changing HMC architectural parameters such as the number of channels can affect the performance of the proposed memory system across different workloads.

2 HETEROGENEOUS HMC+DDRX

Prior research (Kang et al. 2010; Wu et al. 2009) has shown that 3D-DRAM provides significant advantages in terms of performance while enabling energy-efficient computing. 3D-DRAM has a number of superior characteristics, namely high bandwidth, low latency, and low power dissipation. This is achieved by having more parallel accesses to the DRAM enabled by short and fast interconnect. In Table 1, we show the comparison of three emerging memory interfaces using

Table 1. Emerging 2.5D/3D Memory Interface Compared to State-of-the-art DDRx Technology

	LPDDR3	LPDDR4	WIO2	HMC	HBM
Mass Production	2012–13	2014–15	2015	2014	2014–15
Vdd	1.5	1.2	1.2	1.2	1.2
Idd	3.07	2.83	No data	6.64	No data
Bandwidth (GB/s)	17	25.6	51.2	160	128–256
Package Density (GB)	2-4	4-8	4-8	2-4	2-8
Relative Cost per bit	1	1.1	3	2	2
Power Efficiency (mW/GB/s)	67	50	40	35	No data
Power at Max Bandwidth (1GB)	4.61	3.40	No data	7.97	No data

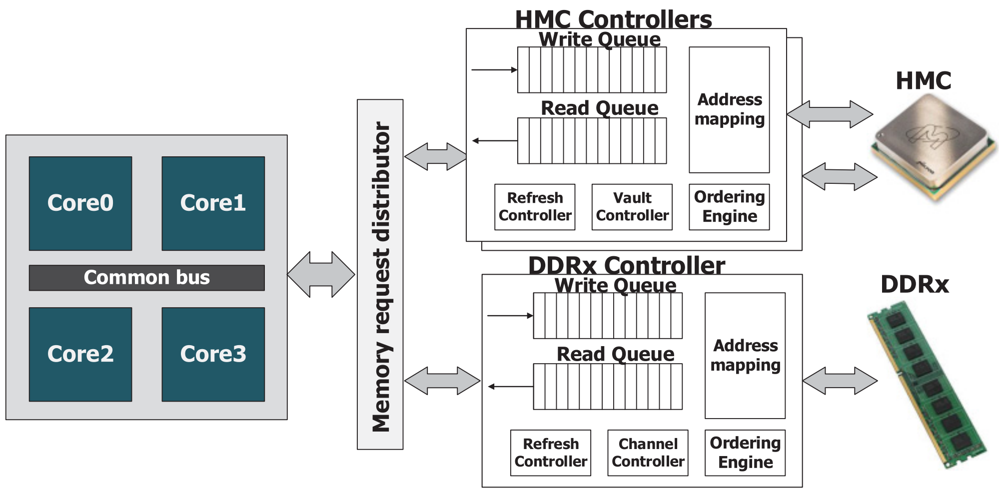


Fig. 2. Studied architecture employing heterogeneous DRAM.

2.5D/3D technology with the state-of-the-art DDR3 and DDR4 interfaces (Pawlowski 2011; Dumasl 2011; Farrell 2012; Jeddeloh and Keeth 2012; Elsasser 2013; Brennan 2013; Jun 2015). As shown, the HMC, Wide I/O (WIO2), and High Bandwidth Memory (HBM) offer much higher bandwidth compared to DDRx technologies. They are also more power efficient. In particular, HMC is superior to DDRx in all those aspects. However, in terms of cost per pit and relative power density (and temperature footprint), DDRx is a better technology (Pawlowski 2011; Jeddeloh and Keeth 2012; Farrell 2012). As a stacked die TSV-based solution, HMC has cost and manufacturing challenges, similarly to HBM and Wide I/O. While the cost might decrease even further in the future, as DRAM is a very cost-sensitive market, DDRx will not disappear any time soon (Elsasser 2013).

Our studied architecture in this work is shown in Figure 2. In our heterogeneous memory system, HMC is combined with a conventional DDRx DRAM to exploit the high memory bandwidth and the low memory latency of the HMC as well as the high capacity and the low cost of the DDRx DRAM. The memory management we employ for the proposed heterogeneous DRAM integrates the OS virtual to physical address translation so the heterogeneous memory is transparent to the CMP (chip multi-processor) and the cores see a unified address space.

As Figure 2 illustrates, the cores memory requests are pushed to the memory request distributor (MRD). Decoding the coming request, MRD transfers the request to the corresponding memory controller (i.e., either HMC or DDRx memory controller). Each controller has its own queue for memory requests. By generating appropriate DRAM commands, the memory controller serves the requests in the queue and accesses the DRAM cells. Then, depending on the request type (i.e., read/write) the data are either written to or read from the memory and sent back to the core through the memory controller's read queue. As shown in Figure 2, our proposed heterogeneous DRAM has two distinct memory channels: one connecting to HMC using two high-speed links and the other connecting to DDRx DRAM. Without loss of generality, similarly to Dong et al. (2010) and Tran et al. (2013), we assume in this article that HMC and DDRx DRAM employ two and one memory controllers, respectively. We also increase the number of HMC channels to study its impact on performance.

The main question for our proposed heterogeneous memory system is how to manage each memory component the HMC and the DDRx DRAM to gain the best performance while addressing bandwidth, capacity, and temperature challenges. The key to answer this question is to understand workloads behavior in terms of memory access pattern and utilization. For instance, the more requests the HMC receives in burst, the more its bandwidth is utilized. However, utilizing the HMC aggressively results in longer memory latency if the workload has a large number of memory requests that are coming in burst. On the other hand, workloads with a large number of memory requests cause more dynamic power dissipation and, thus, higher average temperature. Therefore, a dynamic bandwidth and temperature adaptation is required.

3 HMC+DDRX MANAGEMENT

In this section, we explain our proposed memory management policy. First, we describe our policy to manage the HMC and the DDRx DRAM bandwidth utilization to achieve the best performance in terms of memory access latency. Then, we present our temperature-aware policy to reduce the steady-state temperature rise of HMC while maximizing its performance benefit. It is important to note that the goal of our proposed heterogeneous memory management policy is to distribute the workload requested memory bandwidth to the HMC and the DDRx DRAM.

3.1 Bandwidth Allocation Policy

Memory access latency is a function of memory bandwidth utilization (Dong et al. 2010; Tran et al. 2013). As the bandwidth utilization increases, the memory access latency becomes longer, mainly due to congestion in the memory controller and links. While there are several solutions to mitigate this problem (Kim et al. 2010), above certain bandwidth utilization, due to queuing effect the memory access latency increases significantly (Dong et al. 2010; Tran et al. 2013). In Figure 3, we investigate this phenomenon for both the DDRx DRAM and the HMC independently. As shown Q4 in Figure ??, we increase the bandwidth utilization of HMC and DDRx DRAM by allocating more number of memory requests for each type of studied workloads. The x-axis illustrates the memory request portion that each DRAM receives form the entire accesses. For example, in Figure 3(a), 10/90 means that while 10% of the requests are serviced by the HMC, the rest 90% are serviced by DDRx DRAM. It is important to note that in Figures 3(a) and (b), we show the average memory latency from HMC and DDRx DRAM perspective, respectively. We categorize applications (benchmarks) into three groups; the memory-intensive applications, the memory-non-intensive applications, and a mixture of both. Applications are classified based on their Last Level Cache (LLC) misses per 1K instructions (MPKI) that varies from 0.0005 to 24 for our studied benchmarks. We refer to a application as memory-intensive if its MPKI is greater than 12 and non-intensive if MPKI is less than 1. We create various workloads by combining the memory accesses from applications with different memory intensity behavior. For simplicity, we refer to memory-intensive, memory-non-intensive, and mix workloads throughout the article as MI, MNI, and Mix applications. Workloads used in Figure 3 are representatives of their categories.

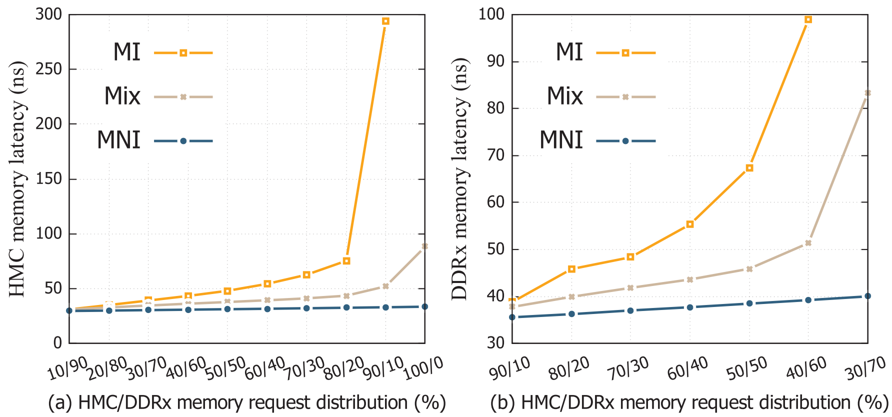


Fig. 3. Memory access latency of (a) HMC and (b) DDRx DRAM as a function of memory request allocation.

As Figure 3(b) shows, for the DDRx DRAM, the MI workload has the highest rise in memory access latency when request allocation increases from 10% to 60% for DDRx DRAM. For bandwidth above 70%, due to the queuing effect the memory access latency for the MI workload becomes so large that we could not show it in the figure (for instance, with 90% utilization the access latency found to be 8,891ns). In Mix and MNI workloads, the memory latency is being affected much less as the bandwidth utilization increases. The results show that for MNI workloads, the memory access latency is somewhat linear, while for Mix applications it grows exponentially but at much slower rate compared to MI workloads. We show the results for HMC in Figure 3(a). Unlike Figure 3(b), for bandwidth above 70%, we are able to present results as MNI and Mix workload access latency were smaller. As shown, when the memory request allocation is between 10% and 80%, the latency is almost linear across all groups of workloads. For larger bandwidth utilizations, except for MNI, in Mix and MI workloads, HMC latency increases exponentially, however, at a much lower rate compared to DDRx DRAM (Figure 3(b)). It is important to notice that, generally, the memory latency increases in DDRx DRAM more quickly compared to HMC, since HMC has a higher memory bandwidth and faster interconnects (TSVs).

Motivated by the observations from Figure 3, we introduce a bandwidth allocation policy to effectively utilize both HMC and DDRx DRAM to gain the minimum average memory access latency for any given workload. In this policy, we allocate new memory pages in an interleaving scheme between HMC and the DDRx DRAM to achieve the minimum average access latency for the entire system. The minimum access latency is achieved at a specific bandwidth utilization of each DRAM, which varies across different workloads. We refer to this point as Optimum Bandwidth Utilization (OBU). For instance, for a given workload, OBU of 60% means that to achieve the minimum access latency we need to distribute the requests to HMC and DDRx DRAM by 60% and 40%, respectively. To satisfy this goal, of 10 new consecutive writes (page faults), we assign the first six access (pages) to the first six free blocks of HMC and the remaining to the four free blocks of the DDRx DRAM. This necessitates a mechanism (using a simple counter) to determine the DRAMs turn. This helps meet the OBU for the new incoming write accesses. Nonetheless, since not all the accesses are new writes (i.e., the requested data already resides in the DRAM), and the access pattern to the previously allocated memory blocks may not be uniform, the target bandwidth allocation might not be satisfied. However, our experimental results show that our memory allocation policy can satisfy the target bandwidth, indicating that the access pattern is somewhat uniform. Table 2 reports the average inaccuracy of our bandwidth allocation technique

Table 2. Average Inaccuracy of Proposed Bandwidth Allocation Policy

Workload	Inaccuracy %	Workload	Inaccuracy %	Workload	Inaccuracy %
MI1	1.3	**Mix1**	0.57	**MNI1**	1.06
MI2	0.13	**Mix2**	0.02	**MNI2**	0.05
MI3	1.12	**Mix3**	0.49	**MNI3**	0.06
MI4	0.16	**Mix4**	0.31	**MNI4**	0.22
Average			0.46		

Table 3. Inaccuracy of Different Request Allocation

	(3,2)	(6,4)	(9,6)	(12,8)	(30,20)	(60,40)
Inaccuracy %	0.46	0.71	0.40	0.52	0.40	0.73

from the OBU for all other target bandwidths (0 to 100 in step of 10), which indicates how accurate it meets the target bandwidth. As reported in Table 2, the average inaccuracy of the proposed allocation policy is 1.32%, that is, reaching close to 99% of the ideal bandwidth utilization.

The main question about the studied inaccuracy is how it can be impacted by different request allocation with the same OBU. For instance, to create 60% of OBU, there are various pairs of HMC to DDRx (HMC,DDRx) request allocations such as (3,2) from 5, (6,4) from 10, (9,6) from 15, (12,8) from 16, (30,20) from 50, and (60,40) from 100 consecutive write requests that might result in different inaccuracies across different workloads. Table 3 reports the average inaccuracy of the aforementioned pairs. As Table 3 presents, the average inaccuracy across all studied workloads is less than 1%, and no specific trend is observed in the results. Therefore, choosing the (3,2) pair is reasonable, as it has the lowest complexity among the possible pairs.

The proposed interleaving memory page allocation policy is shown in Figure 4. As the figure shows, on generating a new request by the CMP, the corresponding core accesses its own TLB and then page table to check whether the address is available in the main memory or not. If so, then, using MRD, the correspondent DRAM is accessed to read/write the data. Otherwise, a page fault occurs, and the bandwidth manager transfers the page that contains the data from the hard disk to a proper DRAM module (i.e., HMC or DDRx). To do so, with the help of OS, the bandwidth manager checks whether any of the DRAMs (i.e., HMC and DDRx) has a free page. If any of the DRAMs is full, then bandwidth manager accesses the one that is not full; otherwise, it employs page replacement policies to bring the new page to the heterogeneous DRAM. Moreover, bandwidth manager needs to know about DRAM's turn to accommodate the new page in the proper DRAM. This is done with the help of the distribution factor variable that stores the OBU. We will discuss temperature-aware distribution factor regulator in Section 3.2.

As discussed, every workload type has a different OBU and the interleaving policy results in the minimum memory latency only if the proper bandwidth utilization is set. Therefore, it is important to detect the type of workload, whether it is a memory intensive, mix or non-intensive, to set the proper OBU. Our studies on workload memory access pattern show that, although the program goes through different execution phases and therefore memory access pattern may changes as a result, consistent with prior work (Kim et al. 2010), the average intensity of memory requests within a given phase is deterministic and highly predictable. Figure 5 illustrates the memory access pattern for two representatives of MI and MNI workloads. The samples are collected every 1 million cycles.

As shown in Figure 5, MNI applications can be clearly distinct from MI workloads, as the number of memory requests in this class of workload remains almost consistently small throughout the 500M cycles studied intervals. Therefore by profiling memory access pattern, we can decide the workload type and the relevant OBU accordingly. As Figure 4 depicts, the memory access

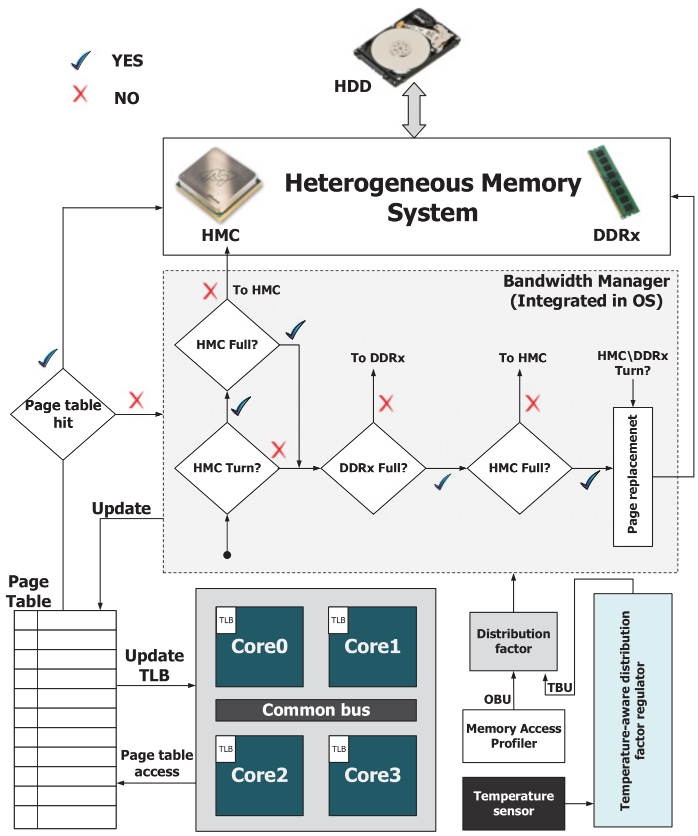


Fig. 4. Bandwidth- and temperature-aware memory management.

profiler provides the proper OBU for the bandwidth manager. This can be done every 10ms, as most operating systems performs context switching at this interval and therefore the memory access pattern will change every interval. After all, as soon as a new page resides in the memory, the corresponding TLB and the page table need to be updated. It is important to note that the access patterns of individual application running on different cores are transparent to the profiler as the profiler resides on the memory controller in the studied heterogeneous architecture. One of the tasks of a memory controller is to differentiate accesses coming from various cores for memory management purposes. Therefore, memory controller is already aware of which accesses coming from which workload (core). Since this is already embedded in the memory controller, our approach does not add overhead for profiling multiple workloads. Therefore, multiple applications create a single workload based on which the profiler decides the intensity class of the accesses during each interval.

Since bandwidth allocation policy brings the memory blocks at a page granularity, and given that we use the same page size as homogeneous memory system does, our memory management does not affect the data locality in DRAM's. Applying our memory allocation policy, we estimate the average memory access latency of the proposed heterogeneous memory system when running different workloads using Equation (1):

$$L_T = (P \times L_{HMC}) + ((1 - P) \times L_{DDRx}), \tag{1}$$

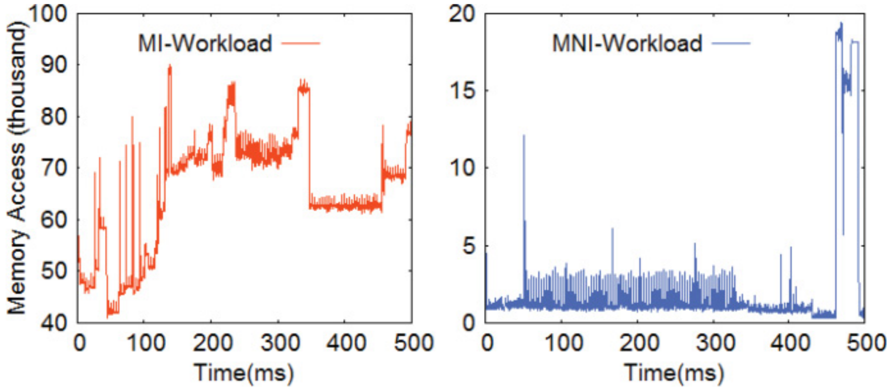


Fig. 5. Memory access pattern for different workloads.

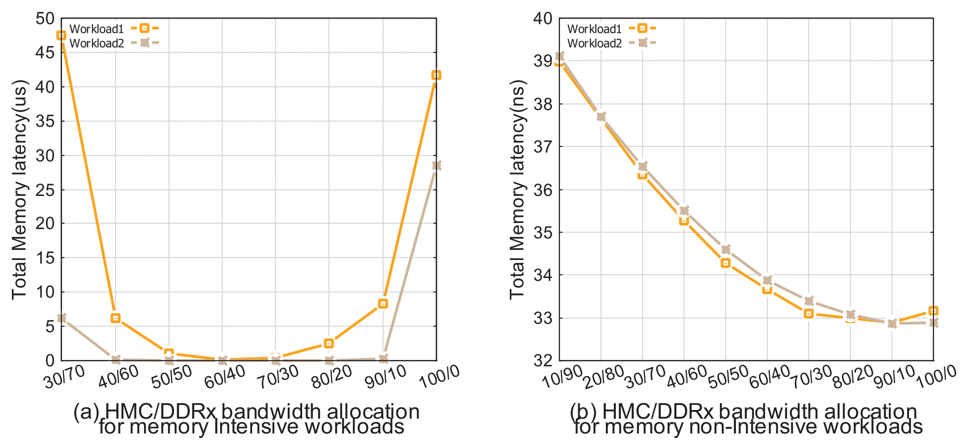


Fig. 6. Total memory access latency in the heterogeneous memory system, as a function of HMC/DDRx bandwidth allocation for (a) MI and (b) MNI workloads.

where LT is the total latency, P is the HMC desired allocated bandwidth, LHMC is the HMC latency, and LDDRx is the DDRx DRAM latency.

Figure 6 presents the total memory access latency for two groups of workloads and for various target bandwidths. It is important to note that memory access latency (Y axis) for MI and MNI is in the range of microseconds and nanoseconds, respectively. We observe such a high difference in memory latency (microseconds vs. nanoseconds) only when the queuing effect occurs in MI workloads. Moreover, as the Mix workload behavior is somewhat close to MI workload behavior, in Figure 6 we only report the results for the first two studied workloads in MI and MNI categories.

As Figure 6 shows, different types of workloads have different OBU to achieve minimum average memory access latency. In MI workloads, the average memory latency is more sensitive to the bandwidth allocation than the other workload. In Figure 6(a), for MI workloads, miss utilization of the heterogeneous memory system results in a large performance loss. For example, for the first workload, if the HMC bandwidth allocation is less than 50% or more than 80%, the memory access latency is becoming large in a microsecond range (note that 50% and 80% of HMC allocation means 50% and 20% of DDRx bandwidth allocation). This occurs for the second workload as well, if the HMC bandwidth allocation is less than 30% or equal to 100%. This large penalty is due

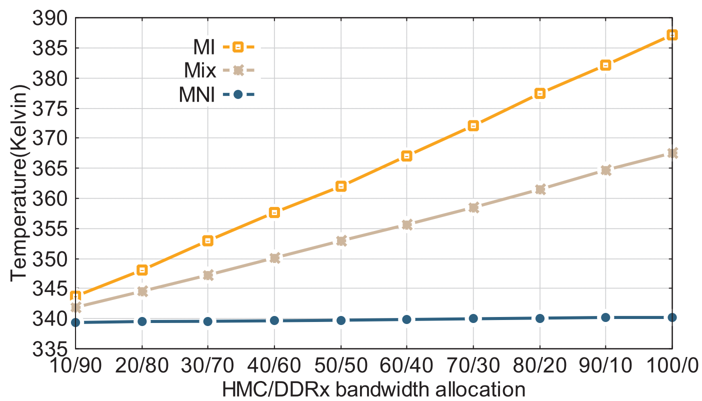


Fig. 7. HMC steady-state temperature of hot spot for various bandwidth allocations across different workloads.

to the queuing effect. It is important to note that as the simulations for both workloads took so long, we were not able to report the memory access latency for 10% and 20% of HMC bandwidth allocation, in Figure 6(a). This shows that the performance loss is even more, compared to 30% of HMC bandwidth allocation. Our observation shows that allocating 60% of the entire bandwidth to HMC results in achieving the best performance for all MI workloads. Therefore, the OBU is set to 60% for this class of workloads. Since Mix workloads show the same behavior as MI workloads do, we set OBU to 60% as well for this class of workloads.

As Figure 6(b) presents, in MNI workload, the performance penalty due to DRAMs miss utilization is very small compared to MI workload. Unlike MI and Mix workloads in which we observed the queuing effect, in memory-non-intensive workloads, as we allocate higher bandwidth to HMC we gain a higher performance up to the point where we reach to 90% of the entire bandwidth. If we allocate the entire bandwidth to HMC, then we lose a small performance. Therefore, we can set the OBU at 90% for this class of workloads. Our observation shows that the average memory access latency of our heterogeneous memory system at the OBU for MI, Mix, and MNI applications are 64ns, 44ns, and 33ns, respectively. It is worth mentioning that the workloads that are not presented in these figures have somewhat similar behavior, and the illustrated workloads can be representative of their corresponding workload category.

3.2 Temperature-Aware Policy

In this section, we propose our algorithm that reduces the steady-state temperature while maintaining the high-performance benefit of bandwidth allocation policy presented earlier. Figure 7 shows the steady-state temperature in HMC as a function of bandwidth allocation, for different types of workloads. As Figure 7 shows, for the MI and the Mix workloads, allocating higher bandwidth to HMC from 10% to 100% results in 25 K and 43 K steady-state temperature increases. For workloads with high memory requests (MI), a sharp rise in temperature is observed when higher bandwidth is allocated. As shown, for the MNI workload, higher bandwidth allocation does not affect the temperature, mainly due to the fact that these workloads do not generate significant memory accesses and therefore they have small power dissipation.

While higher DRAM bandwidth allocation is desired, it comes with a large temperature rise. Such a large thermal rise is not tolerable as it can affect the performance, reliability, and the cooling cost of the design (Kang et al. 2014; Srinivasan et al. 2005). Therefore, we need a smart mechanism to dynamically adapt DRAM bandwidth allocation to manage the temperature.

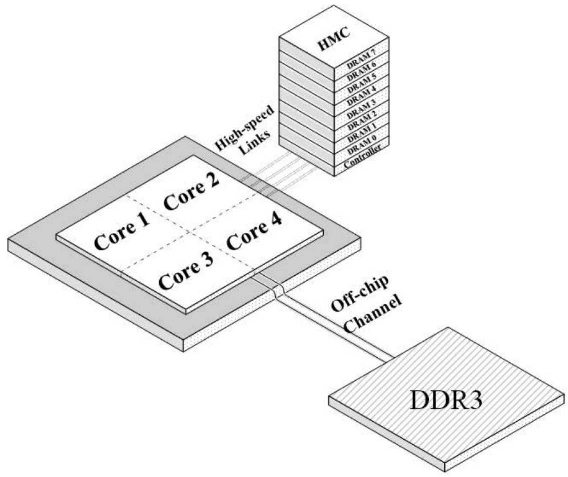


Fig. 8. 3D heterogeneous DRAM combining HMC and DDR3.

3.2.1 Temperature-Aware Bandwidth Allocation. Bandwidth allocation of the heterogeneous DRAM affects its power dissipation. Similarly, the power and therefore the temperature of HMC are highly decided by its bandwidth allocation. As indicated in Figure 7, for MI and Mix workloads there is a large gap in steady-state temperature. Motivated by this observation, we propose our dynamic temperature-aware bandwidth allocation technique (DTBA) to reduce the steady-state temperature of HMC while maintaining high performance benefit.

In DTBA, first we define two operating temperature regions, namely normal and hot. These two regions are separated from each other using the threshold temperature of 78 K. As long as the HMC operates in the normal region it can be utilized to gain the highest performance using the bandwidth allocation policy. However, whenever HMC enters the hot region we allocate it lower bandwidth while dedicating higher bandwidth to the DDRx DRAM at the same time to compensate for potential performance loss. This results in lowering HMC power consumption and therefore reduces steady-state temperature. We implement DTBA using the proposed memory allocation technique explained in Section 3.1 (see Figure 4).

As presented in Figure 7, MNI workloads temperatures are almost bandwidth insensitive. Therefore, these workloads do not require a thermal-aware adaptation, and we can simply use the bandwidth allocation technique to manage their bandwidth utilization.

As Figure 4 shows, our temperature-aware algorithm works as follows. We profile the memory accesses to detect the running workload type. Then, based on the workload type, we set the OBU using the bandwidth allocation policy. The temperature sensor on HMC monitors the temperature periodically. If the HMC temperature rises into the hot region, then the distribution factor variable is over-written with a new bandwidth referred to as Temperature-aware Bandwidth Utilization (TBU). This is done by a temperature-aware distributer factor regulator. Otherwise, we continue with the previous bandwidth allocation based on the OBU provided by the memory access profiler. Our temperature-sampling interval is set at 1ms (Skadron et al. 2003; Zhao et al. 2013).

4 METHODOLOGY

In this section, we explain the framework used to evaluate our proposed heterogeneous DRAM memory page allocation algorithms. As shown in Figure 8, we use a quad-core CMP architecture with a total of 3GBs of DRAM including 1GB HMC and 2GB of DDRx as our target system. We also

Table 4. CMP and Heterogeneous Memory System Parameters

Processor Configuration	
Core Clock	3GHz
Issue and Commit width	4
INT and FP Instruction queue	32 entries
ROB size, INT Reg, FP Reg	128
L1 cache	64KB, 8-way, 2 cycle
L2 cache	512KB, 20 cycle
HMC and DDRx DRAM	
DRAM Clock	800MHz
Column Access Strobe (tCAS)	10 (DDRx), 6 (HMC)
Row Access Strobe (tRAS)	24 (DDRx), 24 (HMC)
Row Buffer Policy	Close page
Page Size	4 KB

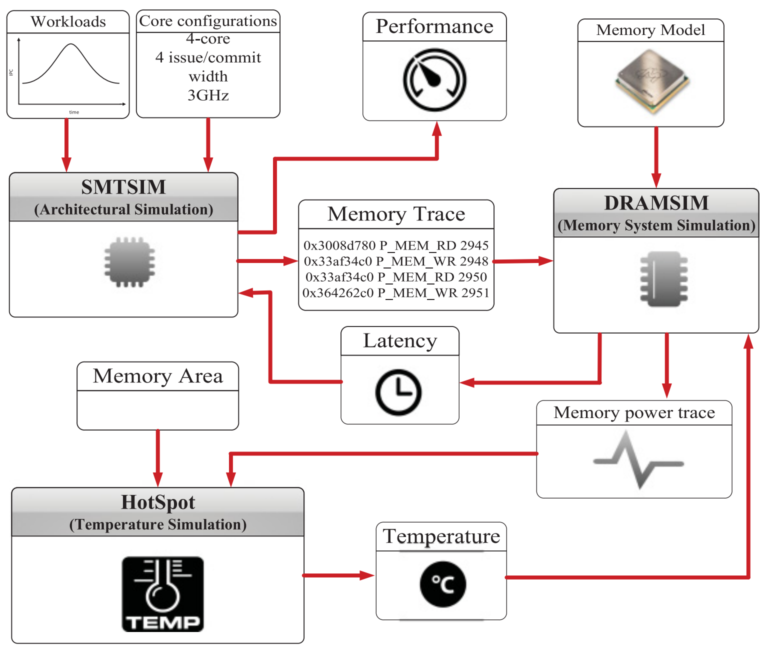


Fig. 9. Framework overview.

increase the number of cores to 8 to study how it affect the OBU obtained by the proposed policies. Moreover, we increase the number of HMC's channel to 4 and 8 to study the impact on DRAM performance in terms of latency. For the DDRx DRAM, we model a Micron DDR3 SDRAM (Rosenfeld et al. 2011). Table 4 summarizes the detailed parameters of CMP architecture and heterogeneous memory system modeled in this work.

Figure 9 gives an overview of our framework. We integrate SMTSIM (Tullsen 1996) and DRAMsim2 (Rosenfeld et al. 2011) simulators for architecture studies. We modify DRAMSim2 memory simulator extensively to model the proposed heterogeneous DRAM. Moreover, DRAMsim2 is equipped with a power profiler to generate the memory subsystem power trace. It is also extended

JETC1401-04 ACMJATS Trim: 6.75 X 10 in July 27, 2017 13:50

Table 5. Thermal Parameters Used in Hotspot

Parameters (A)	**Values**
Die and Interface material thickness	0.05mm
Silicon thermal conductivity	100W/mK
Silicon specific heat	1750KJ/m3K
Spreader thickness	0.01mm
Spreader thermal conductivity	400W/mK
Interface material conductivity	4W/mK
Heatsink thickness	0.01mm
Heatsink conductuvity	400W/mK
Ambient temperature	(45°C)

with a profiler that periodically monitors the memory access pattern to predict whether the workload is memory intensive in the current program phase.

Since the impact of high temperature on neither of leakage and DRAM refresh power are modeled in DRAMsim2, the estimated power of HMC can be inaccurate. As the temperature rises, the leakage current increases, which leads to more leakage power at higher temperature (Li et al. 2009). On the other hand, more leakage power requires the DRAM to be refreshed more frequently. In this work, we take these effects into account to calculate DRAM total power.

To calculate the memory controller power consumption, we use the results reported in Jeddeloh and Keeth (2012). As Jeddeloh and Keeth (2012) presents, in an HMC the average power dissipation of the memory controller is 1.8 of the DRAM cell layers.

As Figure 9 presents, we employ HotSpot (Skadron et al. 2003) to monitor the HMC temperature. To calculate temperature, HotSpot uses power density, which takes into account both the power trace and the area of the chip. In our simulation framework DramSim2 provides the HMC power traces. DRAMs floorplan (area) was adopted from Khurshid and Lipasti (2013).

HotSpot is capable of measuring both transient and steady-state temperature of the chip. It calculates the transient temperature using a thermal-RC-network model (Skadron et al. 2003), and as for steady-state temperature, it calculates the average power using a simpler thermal-R-network model (Zhao et al. 2013; Skadron et al. 2003). In this work, we investigate the affect of our temperature-aware policy on steady-state temperature of the HMC.

Q5 Zhao et al. (2013) has shown that for a majority of standard applications in a multi-core processor, DRAM accesses and thus power consumption is uniformly distributed among DRAM banks. Therefore, as DRAM banks are almost placed on die symmetrically, we assume that the power is distributed evenly across all eight DRAM layers, as well as within each layers. Other studies, including Meng et al. (2011), consider the DRAM temperature to be uniformly distributed as well. We assume the area of the HMC layers including DRAM and controller layers to be 68mm^2, which is adopted from Khurshid and Lipasti (2013). We consider the thickness of the HMC dies and heat-sink to be 0.05mm and 0.01mm, respectively. Other thermal specifications are adapted from Skadron et al. (2003). Table 5 shows the thermal configuration parameters for HotSpot simulation. Similar to Khurshid and Lipasti (2013), since the HMC and CMP are integrated using a PCB, we consider an inexpensive heat-sink for the HMC. As shown in Figure 9, DRAMSim2 receives the transient temperature (running temperature) from HotSpot (Skadron et al. 2003) periodically, that is, every 1ms.

As shown in Figure 9, DRAMSim2 receives the transient temperature (running temperature) from HotSpot (Skadron et al. 2003). This occurs periodically, that is, every 1ms. This feedback

Table 6. Workloads' Benchmarks

Workload type	Workload #	Application
MI	1	cg, mcf, art, applu
	2	sp, lbm_06, gcc_06, em3d_med
	3	applu, art, sp, mcf
	4	lbm_06, gcc_06, cg, em3d_med
Mix	1	perlbmk, gobmk, gzip, h264ref
	2	sp, vortex_3, perlbench_06_diffmail, crafty, namd_06
	3	perlbench_06_splitmail, gobmk, mesa, equake
	4	med, galgel, gap, astar_06_biglakes
MNI	1	perlbmk_makerand, gobmk_06_nngs, gzip_source, h264ref
	2	vortex_3, perlbench_06_diffmail, crafty, namd_06
	3	equake, perlbench_06_splitmail, gobmk_06_trevord, mesa
	4	bisort_med, galgel, gap, astar_06_biglakes

helps the DRAMSim2 to adapt the bandwidth allocation for both HMC and DDRx DRAM for the next interval given the thermal information provided by the HotSpot.

In order to study the performance and thermal characteristics of the proposed heterogeneous memory architecture, we create 24 different workloads, 12 of them for a four-core and 12 others for an eight-core CMP from SPEC2000, SPEC2006, NAS (Bailey et al. 1991) and Olden (Rogers et al. 1995) benchmark suit. Table 6 shows the benchmarks that create the workloads for four-core CMP. For eight-core workloads, we randomly combine four-core benchmarks.

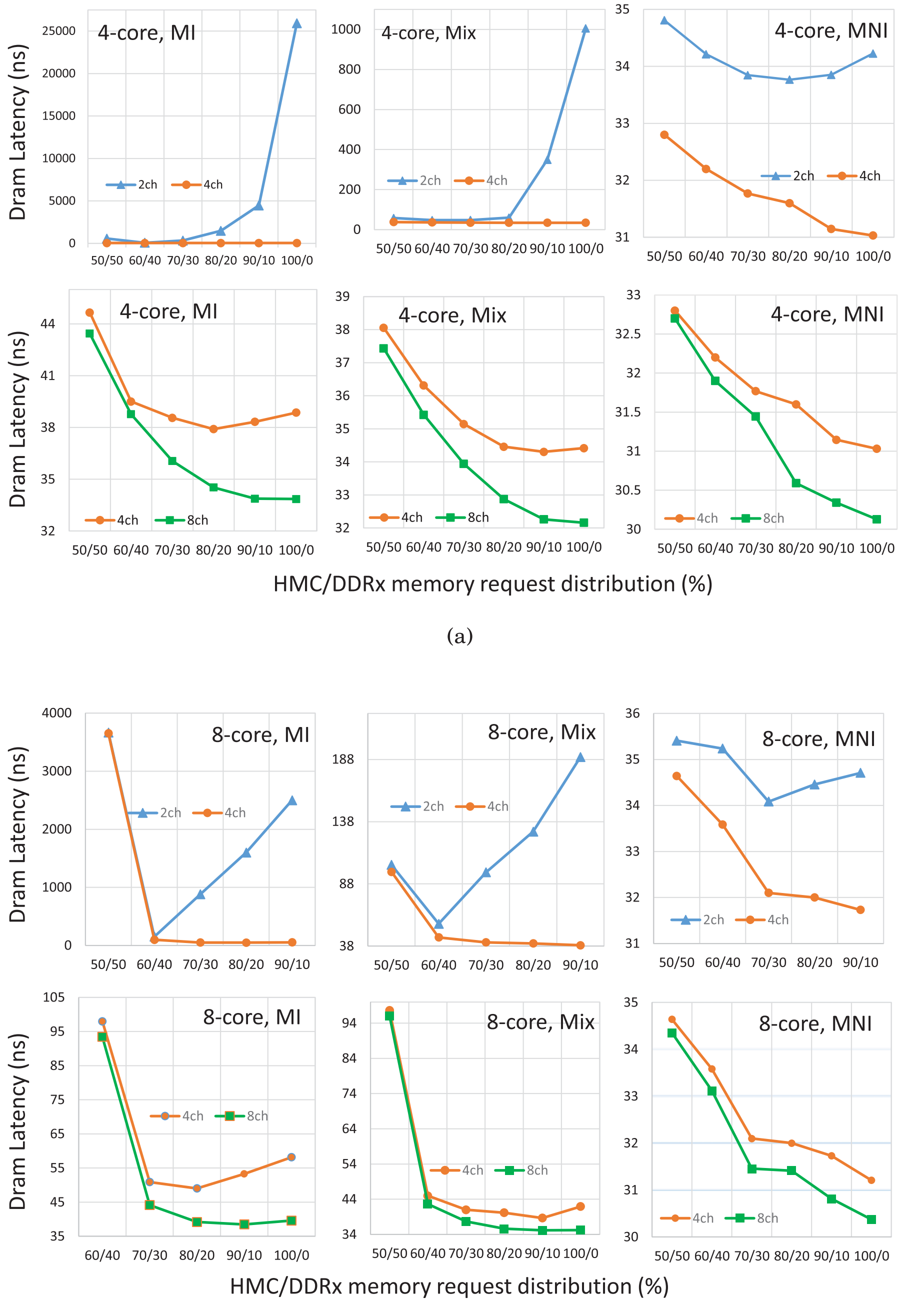
5 RESULTS

In this section, first, we analyze the impact of increasing both the number of HMC's channels and the number of cores on the memory access latency and the OBU. Then we evaluate DTBA when applied in the proposed heterogeneous DRAM subsystem.

5.1 OBU Analysis

Figures 10(a) and (b) show the total DRAM access latency for a four-core and eight-core CMP. Since the memory access latency when the number of channels are 2 is up to four orders of magnitudes larger than when the number of channels are 4 or 8, to show the impact of using more number of channels more clearly, we compare and present the HMC memory latency using two and four channels and then four and eight channels individually.

As shown in Figure 10(a), when a four-core CMP is used, increasing the number of channels from 2 to 4 decreases the memory access latency significantly across both MI and Mix workloads. For instance, this reduces the DRAM access latency from 75ns to 35ns in MI workloads for the optimum bandwith utilization. Moreover, increasing the number of channels from 2 to 4 shifts the OBU from 60% to 80% and 90% in MI and Mix workloads, respectively. Obviously, that is due to the fact that a higher number of channels in HMC results in higher concurrency. Therefore, HMC can serve a larger portion of the total memory requests. By contrast, increasing the number of channels from 4 to 8 comes with a lower improvement in terms of DRAM access latency. That is because the queuing problem has already been resolved by doubling the channels from two to four. As Figure 10(b) illustrates, the impact of increasing channel count from two to four when an eight-core CMP is employed is even higher for MI and Mix workloads. This is due to the fact that using an eight-core CMP puts higher pressure on DRAM in terms of memory request. As a result, using an HMC with more channels is more effective. For example, the DRAM access latency is reduced



(a)

(b)

Fig. 10. DRAM access latency when (a) four and (b) eight cores are used.

Table 7. Optimum Bandwidth Utilization Across Different Platforms and Different HMC Channel Counts

	Four-core			Eight-core		
	2ch	4ch	8ch	2ch	4ch	8ch
MI	60%	80%	100%	60%	80%	90%
Mix	60%	80%	100%	60%	90%	90%
MNI	90%	100%	100%	70%	100%	100%

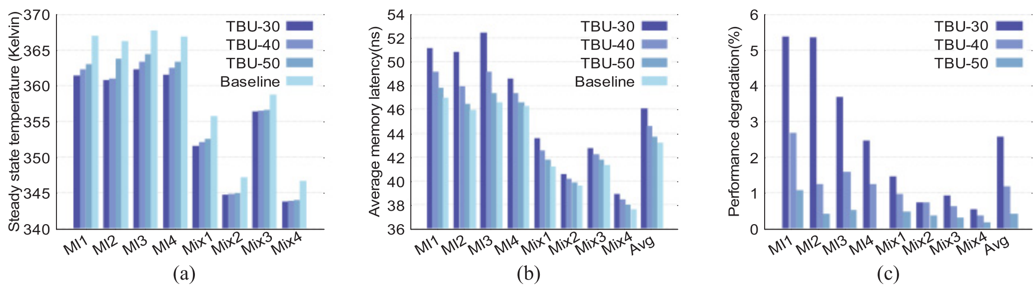


Fig. 11. (a) Steady-state temperature of HMC, (b) average latency of the entire DRAM, and (c) performance degradation for different workloads when different TBU is applied.

from 159ns to 49ns when the number of channels is increased from 2 to 4. Nevertheless, increasing the channel count from four to eight comes with lower improvement in terms of DARAM access latency. Similarly to the four-core case, the OBU is shifted to 80% and 90% across MI and Mix workloads, respectively.

Increasing the number of cores from 4 to 8 results in a higher number of memory requests. However, unlike the MI and Mix workloads, MNI workloads experience only around a 1ns increase in memory access latency due to a higher number of requests when HMC uses two channels. On the other hand, as Figure 7 shows, an HMC with two channels can serve 90% of the requests without facing the queuing problem. Therefore, it is expected that increasing the number of channels results in the lowest performance gain across MNI workloads among all studied workload types. As Figure 10 shows, increasing the channel count from two to eight comes with improvements of less than 3ns and 4ns in DRAM access latency when four-core and eight-core CMPs are used, respectively.

Table 7 reports the OBU for different channel counts across the two studied platforms shown in Figure 10.

5.2 DTBA Results

To find how effective DTBA can optimize temperature and performance simultaneously, we compare it with a performance-optimized (bandwidth allocation) baseline where the bandwidth adaptation is performed to minimize average DRAM access latency and therefore maximize performance. Hence, the OBU is set to 60%, based on the results discussed in Section 3.1. In order to have a better understating of DTBA impact on temperature, we consider different TBU for the hot region discussed in Section 3.2. Figure 11(a) shows the steady-state temperature of DTBA. Note that since MNI workloads are not temperature sensitive, as discussed earlier, only the results for MI and Mix workload are presented.

As shown in Figure 11(a), TBU = 30% configuration achieves the highest temperature reduction. The largest thermal reduction is 5.5 K, which is observed in MI4 workload. TBU = 40% and

TBU = 50% results have slightly lower thermal reduction. Moreover, it is important to note that as memory-intensive workloads are more temperature sensitive, temperature results are more sensitive to the TBU compared to Mix workloads. Since DTBA trades off temperature with performance, it comes with a small performance penalty compared to the bandwidth allocation policy, which is only optimized for performance. This performance loss is due to a longer memory latency. Figures 11(b) and 7(c) show the DTBA performance loss for different workloads in terms of memory latency and IPC.

As shown in Figure 11(b), the average memory access latency increases when DTBA is applied compared to bandwidth allocation policy. Similarly to Figure 11(a), since there is a negligible performance loss for memory-non-intensive workloads, we do not report the results. As Figure 11(b) depicts, for all workloads, the configurations with more temperature reduction result in larger memory latency. The largest increase in average memory latency is observed in MI3 workload. Note that this is the same workload with highest temperature reduction benefit. As Figure 11(c) reports, the average performance loss is around 2.5% in the worst case (TBU = 30%). The loss in performance is more noticeable in MI workloads. This is consistent with the thermal improvement results we show in Figure 11(a) in which a higher temperature reduction is achieved for MI workloads.

6 RELATED WORK

IC designers exploit 3D integration to stack logic on logic (Kontorinis et al. 2014), memory on logic (Meng et al. 2012), and memory on memory (Jeddeloh and Keeth 2012). The main purpose of adopting 3D stacking is to address the technology scaling, cost, performance, power, and energy-efficiency challenges associated with conventional 2D integration both in processor (Kontorinis et al. 2014) and in memory (Jeddeloh and Keeth 2012).

HMC (Jeddeloh and Keeth 2012) and Wide-I/O (JEDEC 2014) are the two state-of-the-art 3D-stacked based DRAM technologies proposed to be used in high-end and low-end computing products.

Prior work on HMC has mostly focused on power or performance management individually (Ahn et al. 2014, 2015b; Han et al. 2014; Zhang et al. 2015). Ahn et al. propose disabling off-chip links of HMC to reduce the leakage power (Ahn et al. 2014, 2015b). Han et al. present a data-aware refresh control technique that dynamically change the refresh rate to suit the distribution of weak cells in HMC (Han et al. 2014). Zhang et al. introduce DLB, a lane-borrowing scheme where lanes are allocated to read and write transmissions dynamically based on the read and write intensity of the application. This results in less contention and thus performance improvement. Moreover, recent research has explored HMC performance thoroughly and compared it against DDRx memory
Q6 system (Rosenfeld). This work also explains and studies the challenges and performance impact of chaining the HMC cubes together.

Also, there has been a number of works exploring the benefits of a generic 3D-DRAM architecture for power and performance. For instance, a generic 3D-DRAM architecture is proposed to be used as the main memory in Kgil et al. (2006). The key idea is to remove the L2 cache so many simple processor cores can be integrated into the same die. The proposed approach then uses 3D-DRAM to provide very high memory bandwidth for the cores. This architecture targets high multi-threading server applications. 3D-DRAM is also proposed to be used as cache and main
Q7 memory in Sun et al. (). In this article, the authors realized that the latencies of large L2 SRAM caches are high, mainly due to the large access latency of Htree. The authors proposed to use TSV to interconnect the processor cores and the caches. This helps the 3D-DRAM cache to be as fast as the SRAM cache. However, as presented in Dong et al. (2010), the performance improvement of using 3D-DRAM as the LLC is not comparable with the performance improvement of using

the heterogeneous memory system. In contemporary systems, it is common that many applications run simultaneously, with different requirements for memory bandwidth and memory access latency (Goossens et al. 2013). Therefore, the performance of the system can be improved if a QoS mechanism is provided. Differentiating application types to provide QoS for homogeneous memory systems is presented in Lin et al. (2003). Our approach differs as it targets heterogeneous memory systems, a more challenging problem in today's complex architecture.

Thermal issues is a main problem that 3D stacking imposes due to the increase in power density. Several studies have attempted to address this issue, particularly focusing on memory-on-logic and memory-on-memory stacking. These studies either propose static methods at design time (Puttaswamy and Loh 2007) or dynamic techniques at runtime (Kang et al. 2014; Meng et al. 2012) to reduce the transient or steady-state temperature. For instance, Kang et al. (2014) proposes a dynamic power and temperature management for a 3D design with stacked cache. Monitoring the runtime application behavior, Meng et al. (2012) attempts to choose the best voltage-frequency setting to achieve the maximum throughput while maintaining the power and temperature constraints in a 3D multicore system with a stacked DRAM. In a recent work, Zhao et al. (2013) proposes a migration technique to reduce temperature in a multicore architecture with stacked DRAM. Migrating threads between cores according to their temperature is the key idea of their work to reduce the steady-state temperature of the system. Another recent work specifically focusing on thermal mitigation of the HMC (Khurshid and Lipasti 2013) attempts to reduce the number of read/write bursts by compressing data in the logic layer (memory controller). This scheme is orthogonal to ours when used in HMC.

To the best of our knowledge, except our short version of this work (Hajkazemi et al. 2015a), no work yet has simultaneously addressed the performance and thermal issues of the HMC. Although Hajkazemi et al. explore Wide-I/O in several aspects, including performance and temperature, the studied target 3D-DRAM is totally different in terms of performance and thermal characteristics from HMC (Hajkazemi et al. 2015b). Moreover, the proposed heterogeneous memory management introduced in Tran et al. (2013) only studies the performance of 3D+2D DRAM. Although this guarantees the quality of service required for an application, it does not investigate the thermal characteristics of the proposed memory system.

7 CONCLUSION

This article proposes an adaptive bandwidth allocation and a temperature-aware memory management to exploit the high bandwidth and low latency of 3D hybrid memory cube (HMC) and high capacity and low temperature of the DDRx DRAM. The bandwidth allocation memory management policy profiles workload at runtime, and, based on that, memory access pattern allocates DRAM and HMC bandwidth accordingly to reduce memory bandwidth congestion. While this ensures high performance, it causes significant thermal rise in HMC. To address this challenge, the temperature-aware policy monitors runtime temperature of HMC to adapt the bandwidth. Temperature-aware policy reduces the temperature while maintaining the high-performance benefit of bandwidth allocation technique. This DTBA technique is done based on application memory access patterns and at runtime. Simulation results show that the bandwidth allocation memory management can utilize the memory bandwidth close to 99% of the ideal bandwidth utilization. Combined with the thermal-aware policy, our proposed memory management reduces steady-state temperature by 4.5 K, on average, across different workloads while maintaining the performance benefits of bandwidth-aware technique.

The bandwidth allocation policy and DTBA work cooperatively to find the target bandwidth that delivers the highest performance while maintaining the HMC temperature below the hot region. Our results show that although allocating 90% of the entire bandwidth to HMC gives the highest performance for memory non-intensive workloads, it hurts performance significantly for memory-intensive and mixed workloads. Therefore, starting with 60% of bandwidth allocation is an optimal choice, as it provides good performance across all workloads.

REFERENCES

Junwhan Ahn, Sungjoo Yoo, and Kiyoung Choi. 2014. Dynamic power management of off-chip links for hybrid memory cubes. In *Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC'14)*. 1–6. DOI: http://dx.doi.org/10.1145/2593069.2593128

Junwhan Ahn, Sungjoo Yoo, and Kiyoung Choi. 2015a. Low-power hybrid memory cubes with link power management and two-level prefetching. (2015). Q8

J. Ahn, S. Yoo, and K. Choi. 2015b. Low-power hybrid memory cubes with link power management and two-level prefetching. *IEEE Trans. VLSI) Syst.* 99 (2015), 1–1. DOI: http://dx.doi.org/10.1109/TVLSI.2015.2420315 Q9

Greg Atwood. 2011. Current and Emerging Memory Technology Landscape. Retrieved from http://www.flashmemorysummit.com/English/Collaterals/Proceedings/2011/20110811_S303_Atwood.pdf.

David H. Bailey, Eric Barszcz, John T. Barton, David S. Browning, Russell L. Carter, Leonardo Dagum, Rod A. Fatoohi, Paul O. Frederickson, Thomas A. Lasinski, Rob S. Schreiber, and others. 1991. The NAS parallel benchmarks. *Int. J. High Perf. Comput. Appl.* 5, 3 (1991), 63–73.

Samta Bansal. 2011. 3D-IC Is Now Real: Wide-IO Is Driving 3D-IC TSV. Retrieved from http://www.flashmemorysummit.com/English/Collaterals/Proceedings/2012/20120821_S101A_Bansal.pdf.

Bob Brennan. 2013. New Directions in Memory Architecture. Retrieved from http://www.memcon.com/pdfs/proceedings2013/keynotes/New_Directions_in_Memory_Architecture.pdf.

Hybrid Memory Cube Consortium. 2014. Hybrid Memory Cube Specification 1.1. Retrieved from http://www.hybridmemorycube.org/specification-download/.

Xiangyu Dong, Yuan Xie, Naveen Muralimanohar, and Norman P. Jouppi. 2010. Simple but effective heterogeneous main memory with on-chip memory controller support. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 1–11.

Sophie Dumasl. 2011. Mobile Memory Forum: LPDDR3 and WideIO. Retrieved from http://www.jedec.org/sites/default/files/Sophie_Dumas_11%2006%20Mobile% %20Memory%20Forum.pdf.

Wendy Elsasser. 2013. 5 Emerging DRAM Interfaces You Should Know for Your Next Design. Retrieved from http://www.chipestimate.com/tech-talks/2013/07/16/Cadence-5-Emerging-DRAM-Interfaces-You-Should-Know-for-Your-Next-Design-. Q10

T. Farrell. 2012. HMC Overview, A Revolutionary Approach to System Memory. Retrieved from http://www.memcon.com/pdfs/keynote_scottgraham.pdf.

Sven Goossens, Jasper Kuijsten, Benny Akesson, and Kees Goossens. 2013. A reconfigurable real-time SDRAM controller for mixed time-criticality systems. In *Proceedings of the 2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS'13)*. IEEE, 1–10.

Marc Greenberg. 2012. How do I decide? Is LPDDR3 or Wide I/O the right memory technology for my next smartphone and tablet? Retrieved from http://www.jedec.org/sites/default/files/Mgreenberg20-20Cadence20-Taiwan_and_Korea_8_17_2012.pdf.

Mohammad Hossein Hajkazemi, Michael Chorney, Reyhaneh Jabbarvand Behrouz, Mohammad Khavari Tavana, and Houman Homayoun. 2015a. Adaptive bandwidth management for performance-temperature trade-offs in heterogeneous HMC+DDRx memory. In *Proceedings of the 25th Edition on the Great Lakes Symposium on VLSI (GLVLSI'15)*. 391–396. DOI: http://dx.doi.org/10.1145/2742060.2742070

Mohammad Hossein Hajkazemi, Mohammad Khavari Tavana, and Houman Homayoun. 2015b. Wide I/O or LPDDR? exploration and analysis of performance, power and temperature trade-offs of emerging DRAM technologies in embedded MPSoCs. In *Proceedings of the 2015 33rd IEEE International Conference on Computer Design (ICCD'15)*. 62–69. DOI: http://dx.doi.org/10.1109/ICCD.2015.7357085

Yinhe Han, Ying Wang, Huawei Li, and Xiaowei Li. 2014. Data-aware DRAM refresh to squeeze the margin of retention time in hybrid memory cube. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'14)*. 295–300. DOI: http://dx.doi.org/10.1109/ICCAD.2014.7001366

Houman Homayoun, Vasileios Kontorinis, Amirali Shayan, Ta-Wei Lin, and Dean M. Tullsen. 2012. Dynamically heterogeneous cores through 3D resource pooling. In *Proceedings of the 18th IEEE International Symposium on High Performance Computer Architecture (HPCA'12)*. 323–334. DOI: http://dx.doi.org/10.1109/HPCA.2012.6169037

Joe Jeddeloh and Brent Keeth. 2012. Hybrid memory cube new DRAM architecture increases density and performance. In *Proceedings of the 2012 Symposium on VLSI Technology (VLSIT'12)*.

JEDEC. 2014. WIDE I/O 2 (WideIO2). Retrieved from https://www.jedec.org/standards-documents/docs/jesd229-2.

Hongshin Jun. 2015. HBM (High Bandwidth Memory) for 2.5D. Retrieved from http://www.semicontaiwan.org/zh/sites/semicontaiwan.org/files/data15/docs/4_5_semicon_taiwan_2015_ppt_template_sk_hynix_hbm_r5.pdf.

Kyungsu Kang, Giovanni De Micheli, Seunghan Lee, and Chong-Min Kyung. 2014. Temperature-aware runtime power management for chip-multiprocessors with 3-D stacked cache. In *Proceedings of the 2014 15th International Symposium on Quality Electronic Design (ISQED'14)*. IEEE, 163–170.

Uksong Kang, Hoe-Ju Chung, Seongmoo Heo, Duk-Ha Park, Hoon Lee, Jin Ho Kim, Soon-Hong Ahn, Soo-Ho Cha, Jaesung Ahn, DukMin Kwon, and others. 2010. 8 Gb 3-D DDR3 DRAM using through-silicon-via technology. *IEEE J. Solid-State Circ.* 45, 1 (2010), 111–119.

Taeho Kgil, Shaun D'Souza, Ali Saidi, Nathan Binkert, Ronald Dreslinski, Trevor Mudge, Steven Reinhardt, and Krisztian Flautner. 2006. PicoServer: Using 3D stacking technology to enable a compact energy efficient chip multiprocessor. In *ACM Sigplan Notices*, Vol. 41. ACM, 117–128.

Mushfique Junayed Khurshid and Mikko Lipasti. 2013. Data compression for thermal mitigation in the hybrid memory cube. In *Proceedings of the 2013 IEEE 31st International Conference on Computer Design (ICCD'13)*. IEEE, 185–192.

Yoongu Kim, Dongsu Han, Onur Mutlu, and Mor Harchol-Balter. 2010. ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers. In *Proceedings of the 2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA'10)*. IEEE, 1–12.

Vasileios Kontorinis, Mohammad K. Tavana, Mohammad H. Hajkazemi, Dean M. Tullsen, and Houman Homayoun. 2014. Enabling dynamic heterogeneity through core-on-core stacking. In *Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC'14)*. IEEE, 1–6.

Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009. (MICRO-42'09)*. IEEE, 469–480.

Tzu-Chieh Lin, Kun-Bin Lee, and Chein-Wei Jen. 2003. Quality-aware memory controller for multimedia platform SoC. In *Proceedings of the IEEE Workshop on Signal Processing Systems, 2003 (SIPS'03)*. IEEE, 328–333.

Jie Meng, Katsutoshi Kawakami, and Ayse K. Coskun. 2012. Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. In *Proceedings of the 49th Annual Design Automation Conference*. ACM, 648–655.

Jie Meng, Daniel Rossell, and Ayse K. Coskun. 2011. Exploring performance, power, and temperature characteristics of 3D systems with on-chip DRAM. In *Proceedings of the 2011 International Green Computing Conference and Workshops (IGCC'11)*. IEEE, 1–6.

Li Ming. 2013. 3D Packaging for Memory Application. Retrieved from http://www.avsusergroups.org/joint_pdfs/2013_6Li.pdf.

J. Thomas Pawlowski. 2011. Hybrid memory cube: Breakthrough DRAM performance with a fundamentally re-architected DRAM subsystem. In *Proceedings of the 23rd Hot Chips Symposium*.

Kiran Puttaswamy and Gabriel H. Loh. 2007. Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3d-integrated processors. In *Proceedings of the IEEE 13th International Symposium on High Performance Computer Architecture, 2007 (HPCA'07)*. IEEE, 193–204.

Anne Rogers, Martin C. Carlisle, John H. Reppy, and Laurie J. Hendren. 1995. Supporting dynamic data structures on distributed-memory machines. *ACM Trans. Program. Lang. Syst.* 17, 2 (1995), 233–263.

Q11 Paul Rosenfeld. *Performance Exploration of the Hybrid Memory Cube*. Ph.D. Dissertation. University of Maryland at Collage Park.

Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. 2011. DRAMSim2: A cycle accurate memory system simulator. *Comput. Arch. Lett.* 10, 1 (2011), 16–19.

Kevin Skadron, Mircea R. Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and David Tarjan. 2003. Temperature-aware microarchitecture. *ACM SIGARCH Comput. Arch. News* 31, 2 (2003), 2–13.

Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, Jude Rivers, and others. 2005. Lifetime reliability: Toward an architectural solution. *IEEE Micro* 25, 3 (2005), 70–80.

Q12 Hongbin Sun, Jibang Liu, Rakesh Anigundi, Nanning Zheng, Jian-Qiang Lu, Ken Rose, and Tong Zhang. Design of 3D DRAM and its application in 3D integrated multi-core computing systems.

Hossein Tajik, Houman Homayoun, and Nikil Dutt. 2013. VAWOM: Temperature and process variation aware wearout management in 3D multicore architecture. In *Proceedings of the 50th Annual Design Automation Conference 2013 (DAC'13)*. 178:1–178:8. DOI: http://dx.doi.org/10.1145/2463209.2488953

Le-Nguyen Tran, Fadi J. Kurdahi, Ahmed M. Eltawil, and Houman Homayoun. 2013. Heterogeneous memory management for 3D-DRAM and external DRAM with QoS. In *Proceedings of the 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC'13)*. IEEE, 663–668.

Dean M. Tullsen. 1996. Simulation and modeling of a simultaneous multithreading processor. In *Proceedings of the 1996 22nd International Conference for the Resource Management & Performance Evaluation of Enterprise Computing Systems (CMG'96)*. 819–828.

Qi Wu, Ken Rose, Jian-Qiang Lu, and Tong Zhang. 2009. Impacts of though-DRAM vias in 3D processor-DRAM integrated systems. In *Proceeedings of the IEEE International Conference on 3D System Integration, 2009 (3DIC'09)*. IEEE, 1–6.

Tao Zhang, Cong Xu, Ke Chen, Guangyu Sun, and Yuan Xie. 2014. 3D-SWIFT: A high-performance 3D-stacked wide IO DRAM. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*. ACM, 51–56.

Xianwei Zhang, Youtao Zhang, and Jun Yang. 2015. DLB: Dynamic lane borrowing for improving bandwidth and performance in hybrid memory cube. In *Proceedings of the 2015 33rd IEEE International Conference on Computer Design (ICCD'15)*. 125–132. DOI: http://dx.doi.org/10.1109/ICCD.2015.7357093

Dali Zhao, Houman Homayoun, and Alexander V. Veidenbaum. 2013. Temperature aware thread migration in 3D architecture with stacked DRAM. In *Proceedings of the 2013 14th International Symposium on Quality Electronic Design (ISQED'13)*. IEEE, 80–87.

Received November 2011; revised March 2017; accepted May 2017

Author Queries

Q1: AU: Please define 3D-DRAM and DDRx at first occurrence in the abstract and in the main text.

Q2: AU: Please provide full mailing and email addresses for all authors.

Q3: AU: Please define QoS at first occurrence.

Q4: AU: In Section 3.1, please fix coding so the correct figure is called out (it appears as the double question mark).

Q5: AU: Sentence “Zhao et al. [2013] hsa shown that for...”: please review and reword for clarity.

Q6: AU: Please fix the code for this reference by including the year.

Q7: AU: Please fix the text and code for the Sun et al. reference; please include year.

Q8: AU: Ahn 2015a: Please update and complete as per style.

Q9: AU: Ahn 2015b: please provide missing issue or volume number.

Q10: AU: Elsasser: URL as meant? Hyphen at the end as meant?

Q11: AU: Rosenfeld: please add year to reference text and to code so it appears in the text cite as well.

Q12: AU: Sun et al: Please update and complete as per style.